The Hidden Costs of Open Source and Homegrown Applications

The Economics of Enterprise Software

by Julian Fish, Director of Products, Serena Software (now Micro Focus)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Release Control: Why Is This Different?</td>
<td>1</td>
</tr>
<tr>
<td>Accounting for Real Costs</td>
<td>2</td>
</tr>
<tr>
<td>When 'Free' Is Not Really 'Free'</td>
<td>2</td>
</tr>
<tr>
<td>Other Considerations</td>
<td>3</td>
</tr>
<tr>
<td>The Problem of “Sparseness”</td>
<td>3</td>
</tr>
<tr>
<td>Real TCO: A Complex Algorithm</td>
<td>5</td>
</tr>
<tr>
<td>A Pragmatic Approach</td>
<td>7</td>
</tr>
<tr>
<td>Sources of Savings</td>
<td>9</td>
</tr>
<tr>
<td>Comparing the Costs</td>
<td>11</td>
</tr>
<tr>
<td>Conclusion and Summary</td>
<td>12</td>
</tr>
<tr>
<td>Appendix A: Comparing Open Source versus Micro Focus Release</td>
<td>13</td>
</tr>
<tr>
<td>Control TCO</td>
<td></td>
</tr>
</tbody>
</table>
Introduction

The willingness of enterprises to adopt commercial and non-commercial open source software has led many CIOs and CTOs to challenge investment in commercial off the shelf (COTS) software. However, many organizations who investigate the value and real world expense between open source and commercial offerings uncover significant unexpected costs with open source tools. These costs can raise exponentially when home grown solutions are in turn created on top of open source products. While ‘free’ is certainly an attractive number for CEOs, CTOs and CIOs, there remains a misconception over the true underlying costs of open source adoption. These costs are explored in this paper in order to give a true cost vs. value perspective. We will also examine how the Micro Focus® Release Control solution compares favorably to building a similar solution based on either open source software or other BPM platforms.

Release Control: Why Is This Different?

The pros and cons of open source versus commercial software have been debated in corporate IT for years. Despite the many benefits of open source software, it is not without its pitfalls. While originally viewed as a decision between Windows and Linux, new product lines, capabilities and market requirements have meant that open source tools can now address a large number of organizational challenges. For example, the prevalence of open source tools in enterprise IT for defect tracking, source code management and build management frequently leads to questions by highly skilled research and development staff as to the value of COTS applications. In the case of open source products, where use cases and implementation approaches are common between teams, departments and organizations, it is easy to justify and accept that they are both viable and cost effective options. However, with a domain as organizationally challenging as Release Management, where use cases, working practices and ongoing process improvement are key, there are a number of additional factors that must be taken into consideration. Creating a solution for release management is certainly not a trivial undertaking. This is particularly true for organizations that may lack staff with requisite expertise to operate and maintain open source software and the software applications that they develop and build on the open source platforms. Indeed, it has been the experience of Micro Focus that even customers experienced with the powerful SBM BPM process engine struggle to implement a truly successful release management offering.
Accounting for Real Costs

Despite the many benefits of open source software, it is not without considerable pitfalls and shouldn’t be seen as a panacea for all organizational ills. This is especially true for organizations that are technically competent, but who lack staff with the requisite expertise to develop and maintain both open source and open source derived software, support its infrastructure requirements and build cross-system integrations. Similarly, organizations who believe that they can simply ‘roll their own’ release process model frequently underestimate the complexities of both the integrated nature of a release control tool-chain and the level of dependencies across multiple applications.

When ‘Free’ Is Not Really ‘Free’

The costs associated with deploying and maintaining open source software can manifest themselves in several ways. These may include:

- Increased time spent on system administration
- Excessive time spent troubleshooting problems due to the lack of a formal support channel
- Costs related to handling functions such as quality assurance and regression testing in-house
- Reduced productivity due to system downtime and a lack of working knowledge on how to repair issues
- Additional expenses in education if new software development skills are required to maintain an open source environment
- Creeping operational costs as organizations find themselves not just in the application research and development business, but also in the software build, documentation and maintenance business
- Unexpected development and maintenance costs that can arise when needed functionality cannot be fulfilled ‘out of the box’ by available open source software or that will take significant investment to build based upon open source technology
- Organizational risk as key developers understand vital applications: staff attrition can lead to unmaintainable systems or increased retention costs
- Reduction in available resources as team members focus more on keeping systems running and less on performing their ‘day job’
Other Considerations

Aside from the factors above, several other items can contribute to cost and risk when either directly using or building applications based on open source software. These may include:

- Lack of technical roadmaps for open source software can make planning a challenge.
- Lack of upgrade or new feature development may leave organizations vulnerable to security issues.
- Important software maintenance such as performing updates and applying security patches may be avoided for fear of breaking fragile open source based software environments.
- Open source software choices may preclude the selection of new hardware or compatible software, constraining options and leading to costly re-engineering efforts downstream.
- The quality of open source community support varies and, if available at all, is often narrowly focused on specific software components. This lack of community support leaves administrators to shoulder the burden of software integration issues themselves. This is especially true when applications are built or extended upon using open source software.
- Open source software is frequently targeted at specific use cases. In the event that specific integrations must be performed, organizations are required to build them. Finding the right resources and skills to understand system integrations between Windows, Linux, UNIX and mainframe environments and technologies is often difficult if not impossible.
- Commercial applications are generally stored in ESCROW, ensuring that if the vendor were to cease trading, all sources can be attained and software recreated. With applications based upon open source technology, the application source and the application itself are typically stored in the development teams’ version control repository. Without secure storage, there is a risk that the software source could be accidentally or maliciously removed with no way to restore it, making it very difficult for the software to be maintained and continued for future releases.

The Problem of “Sparseness”

Comparisons between open source and commercial software often assume that the alternatives have equal technical merit. This may hold true for components in widespread use like operating environments, databases and scripting languages where the quality of open source software is well recognized. It is often not true, however, for more specialized software that is less widely deployed and less thoroughly exercised.
In certain specialized areas, open source products may not exist at all. If they do exist, they may provide less functionality, add more complexity and require more effort to adapt and install than their commercial counterparts.

A simple example of this would be task sequencing from within a release process. When performing release activities, a number of tasks will often need to be performed either in sequence or in parallel. Having an application with built-in sequencing capability, with dependency management and task ordering, will enable users to support highly complex yet interdependent deployments. Supporting both manual and automated tasks is key to successful deployment, audit and error logging.

Again, many customers who have attempted to implement such basic yet critical functionality in the SBM platform while ‘rolling their own’ have failed to deliver this critical interaction.

Another key area of interaction that many organizations wish to align is the process of releasing to mainframe and distributed systems simultaneously. Although sounding simple in theory—how difficult can it be to get technology from the ’60s to communicate nicely with technology from the ’80s—the reality of the level of complexity is only understood at implementation time.

Open source solutions can often face challenges when reporting, trying to gain visibility into real-time status or proving compliance-relevant information. Some specific areas where open source solutions can be challenging to find or successfully integrate include:

- Reporting and analysis tools
- Integrations with engineering workflow tools
- Integrations with application development tools and application development workflow tools
- Workload driven provisioning solutions
- Integrations with deployment automation technologies
Many of these tools are asynchronous in nature and require either complex queries to glean status or the emission and reception of API, SOAP or REST events. Creating a listener for a number of application interfaces soon becomes a full-time occupation.

These and other capabilities often emerge as critical requirements. However, because open source solutions are unavailable or lack needed features, organizations can face significant costs to develop, purchase or integrate the needed functionality.

Real TCO: A Complex Algorithm

The decision on whether to deploy open source or commercial software for release management is sometimes seen as a binary choice. In practice, organizations have a range of different options.

Figure 1 on the following page illustrates a range of alternatives between open source and commercial solutions. The shape of the total cost of ownership (TCO) curve will vary depending on the environment, so this is not meant to be definitive, but rather to illustrate that different organizations will weigh costs and benefits differently.

For most organizations, being at one extreme or the other is likely to be expensive and limits their options and downstream flexibility.
Organizations that deploy and support their own pure open source environments (operating at the left end of the diagram in Figure 1) take on a significant amount of in-house integration, development and support cost. For organizations that have deep technical expertise and are already developing applications, this may be costly but manageable. But for smaller organizations, this can be prohibitively expensive.

At the other extreme, with a pre-defined commercial solution installed on a proprietary operating environment, like Windows Server, costs arise in different places. Organizations will pay higher costs for software licensing, maintenance and professional services, but they will enjoy a better-supported, easily upgradable and integrated system.
A commercially supported release control product integrated with their choice of other technologies, proprietary or otherwise, provides users with the “best of both worlds.”

A risk that can impact release management tools in Windows environments is lack of access to the vast number of open source release management tools that exist on Linux or UNIX. Organizations operating a fully proprietary environment can incur additional costs as they are forced to purchase or develop needed solutions themselves.

There can also be surprises related to personnel costs. For example, it cannot be assumed that the same skills suitable for administering a Java or .NET environment are readily portable to the more complex task of building and administering an open source release management system. Regardless of the environment deployed, deeper expertise impacting personnel costs will be required.

**A Pragmatic Approach**

Many customers opt for a more pragmatic approach, blending open source and commercial software in a fashion that minimizes total costs while supporting the full range of applications that they expect to run. A commercially supported release control product integrated with their choice of other technologies, proprietary or otherwise, provides users with the “best of both worlds.”

Using this combination of products provides freedom of choice while also reducing operation and support costs. The pre-packaged release control solutions give the advantage of a rich set of pre-defined installation, upgrade, and scalability capabilities and offer ease of customization using a graphical process designer. Users do not have to deal with the challenge of writing integrations to open source systems, as these typically already exist. In addition, customers have access to a technical support organization to help resolve problems quickly.

Micro Focus release control provides a pre-packaged product that specifically occupies this TCO “sweet spot,” as illustrated in Figure 2 on the following page.
Deployment pipelines have existed in many guises over the years, from the traditional SDLC where development is followed by test followed by production, to visually defined and dynamic paths to production that may vary depending upon the application and potential risk of deploying said application.

Micro Focus Release Control is a commercially available, customizable process management engine.

To help ensure that this goal is achieved, the notion of a deployment pipeline or path to production is key. Deployment pipelines have existed in many guises over the years, from the traditional SDLC where development is followed by test followed by production, to visually defined and dynamic paths to production that may vary depending upon the application and potential risk of deploying said application.
Sources of Savings

Deploying a release control platform can help organizations save or avoid costs in several ways, including:

- Reduced administration
- Reduced training and support costs
- Improved user productivity
- Reduced implementation time
- Pre-defined best practice processes
- Avoidance of unanticipated costs

The following sections examine some of these sources of savings in more detail. Personnel savings: Even for experienced Software developers, building a fully functional release control solution from open source components can take many weeks. Micro Focus Release Control will dramatically reduce the time to install and configure an organization’s release processes because it is pre-certified, pre-tested, includes best practice process definitions, and comes with many pre-configured integrations. Because of its comprehensive, easy-to-use web interface, Micro Focus Release Control can be easily installed and managed by non-specialists. The graphical composer tool simplifies process changes and system-to-system interactions.

Improved productivity: Micro Focus Release Control is designed to be easy to understand, scalable and fault tolerant. This ensures that system downtime is kept to a minimum. Features like repository promotions and process snapshots take the risk out of tasks like software upgrades and patching. They enable administrators to easily roll back to a known good configuration if anything goes wrong with a software update or patch installation. This helps reduce downtime, makes software changes simple and provides peace of mind to users concerned about breaking functionality by performing a software update.

Hidden cost avoidance: Some organizations do not include requirements gathering, story definition, QA and training overhead as part of the cost of their open source deployment. Acquiring requirements, building integrations and validating these tools can result in significant costs. Many other release management solutions require additional expenditures for many of the capabilities that are included as standard features in Micro Focus Release Control. Examples of these extra-cost components may include:
Notification and scheduling capabilities
Work or task management
User-centric application-aware web UI
Commercial grade scalability
Rules-based processing engine
Graphical process modeling capabilities
Process packs

Pre-defined best practice processes: There is a significant difference in the cost and productivity of an organization running at 80 percent utilization versus one running at 95 percent utilization. By taking advantage of superior scheduling capabilities, visibility, insight and repeatability, all based upon industry proven processes, customers can better align resources to business needs and can achieve better levels of organizational utilization, allowing them to do more. With a do-it-yourself approach to building products from open source components, customers risk deploying un-optimized processes and toolset integrations. The components may work, but at reduced performance levels, undermining the whole purpose of deploying a release control solution.

Reduced risk: By relying on pre-tested, certified configurations fully backed by Micro Focus, customers are assured that any issues can be addressed quickly and efficiently, typically without the need for on-site consultants or additional support expertise that result in un-budgeted costs.

Avoidance of unanticipated costs: Many sites run both development and production release control instances. Rather than provisioning separate clusters, total costs can be reduced by using Micro Focus Release Control to manage a system with development and production processes. Micro Focus Release Control provides process promotion as a standard feature, allowing customers to use a single tool to manage development and production processes while avoiding unnecessary infrastructure spending.

“Future-proofing”: Over time, sustaining costs tend to dominate IT-related spending. An application might run for a few years, but when it comes time to expand or modify, upgrades can be a challenge since new hardware platforms may require new operating systems and drivers that did not exist on the market when the application was originally installed.
Micro Focus maintains and actively supports new operating systems, drivers and security patches, ensuring platform currency without the need to re-install the application from scratch. This means that processes and infrastructure can be grown incrementally, avoiding costly downstream “rip and replace” scenarios and extending the useful life of the software.

Comparing the Costs

TCO estimates vary based on many factors including the nature of the installation, inhouse capability, types of applications to be deployed and cost of down-time. For example, consider a typical implementation supporting ten release managers who each spend roughly 30 percent of their time working with Micro Focus Release Control. For the purpose of this analysis, it is assumed that the environment is supported by a single, fully competent part-time administrator and that the customer will deploy the same processes, integrations and resources to configure the initial environment and use the same operating system.

Fig. 3

Initial cost of ownership of Release Control vs Open-Source
Figure 3 (previous page) compares the annual TCO for this hypothetical environment when running open source software with the same environment running a commercially supported, fully integrated Micro Focus Release Control environment. As the figure illustrates, organizations not only need to make their own determinations of TCO based on how they account for details such as the cost of labor and depreciation, but they also need to include productivity costs. Because a non-productive or idle engineering team is vastly more expensive than a part-time administrator, these costs can rapidly dwarf administration costs. Therefore, the real opportunities for savings lie in areas such as application integrations, which drive productivity, and timely risk-free upgrades, which reduce down-time and improve resource utilization.

Appendix A details the assumptions and calculations behind the TCO estimates in Figure 3. If the example depicted an environment with more active users or less in-house open source development expertise, the argument for commercial software would become even more compelling. The model assumes that there was an opportunity to increase the deployment rate by 10 percent through more effective processes, utilization of a common tool, or more sophisticated resource sharing. Even if there were no process improvement cost savings, reductions in administrative costs and positive impacts on user productivity make the case for Micro Focus Release Control compelling. In addition, the useful lifespan of an application was not accounted for. The lifespan might be extended in a dynamic environment by simplifying processes over time, further improving cost savings.

**Conclusion and Summary**

While open source applications are prevalent in many R&D organizations, there are many issues related to cost and development complexity that can make open source solution implementation challenging. In addition, determining real costs can be complex because every environment is different, and organizations will assess costs using their own methodologies and based on their own requirements and capabilities.

For many organizations, the most practical and cost-effective approach is to deploy a commercially supported release management product on a maintained and expandable platform. Micro Focus Release Control represents an attractive solution because it is a complete product that combines rapid ROI, low TCO, ease of use, integration capabilities and rapid deployment while avoiding the headaches and management costs of open source software.
Because it is a complete, integrated product, Micro Focus Release Control allows analysts, developers, release engineers and executive staff to focus on their work, glean visibility of performance metrics and identify process bottlenecks and overheads, rather than on less productive pursuits such as building, deploying, managing and supporting open source tools or applications.

**Appendix A: Comparing Open Source versus Micro Focus Release Control TCO**

Due to the diverse nature of organizational release processes, full ROI and comparison information is available from Micro Focus Software. The numbers used in this report are indicative of averages found in the past five years.

**Additional Information**

- Reductions in infrastructure support and power costs are driven by assumption (u).
- Unforeseen incremental costs (v) stem from the cost of addressing functional requirements that open source software may not readily support. These costs may take the form of software purchases, consulting time, or time and effort to find, integrate, deploy and support additional layered solutions.
- This analysis assumes a 220 day working year—for example 22 days of effort (a little over four working weeks) would be considered to be 10 percent of an individual’s time when accounting for cost.
- There are costs related to administration and user-productivity regardless of whether open source or commercial software is chosen. Therefore the model only accounts for incremental costs that may manifest themselves in open source deployments. Similar costs associated with Release Control are assumed to be baseline costs.
- This analysis is simplified for this whitepaper, but is based on a TCO calculator developed by Serena Software Europe Ltd. (now Micro Focus)

**For More Information**

To learn more about Micro Focus Release Control, please contact your Micro Focus sales representative or Micro Focus Business Partner, or visit: [www.microfocus.com](http://www.microfocus.com)