Cost Conscious:

A practical guide for understanding and calculating the financial benefits of open source for enterprise IT projects

451 COMMERCIAL ADOPTION OF OPEN SOURCE (CAOS) RESEARCH SERVICE

REPORT 2, NOVEMBER 2006

the 451 group

451 COMMERCIAL ADOPTION OF OPEN SOURCE (CAOS) RESEARCH SERVICE NOVEMBER 2006

COST CONSCIOUS:

A practical guide for understanding and calculating the financial benefits of open source for enterprise IT projects

Abstract: The promise of open source software in enterprise IT has been strongly linked to the potential for cost savings. While reasons for open source adoption vary, achieving cost savings continually ranks as one of the top motivators – yet the tools to track these potential savings are not always available. Like any technology decision – and especially given the continued cost pressure placed on IT budgets – the adoption of open source requires a business justification, with cost playing an important role in this decision.

This report serves as a practical guide for understanding and calculating the financial benefits of open source. It will introduce IT managers and architects to the basics of financial analysis – concepts, processes and elements – and will provide a tool to help identify and capture the costs and potential benefits for adopting open source software.

KEY FINDINGS

- Lower cost remains one of the top reasons for open source software adoption. We do not believe that cost savings as a major motivator for open source software adoption is going away anytime soon. This has implications for open source software vendors, and how they sell into organizations. This is especially true for subscription-based open source software vendors that are not always the low-cost option. Open source alternatives without a compelling cost savings component will likely face hurdles with enterprise IT buyers.
- Saving money on software licenses is seen as the greatest opportunity for cost savings, even though licenses often represent only a fraction of the total cost of an IT project over its lifetime. As the cost savings for software licenses tend to be more dramatic for new projects or large-scale growth of existing projects, this raises some issues relating to smaller-scale projects or revisions to existing open source deployments, once the major source of cost savings is gone. This kind of short-term focus may harm open source adoption in some cases.
- An organization often begins its exploration of open source with the goal of obtaining financial benefits, only to find that other benefits may well outweigh the financial objectives. After the adoption of open source software, the primary benefit shifts to flexibility. We see cost savings as opening doors for open source adoption, while flexibility is the long-term benefit. Open source software vendors and project teams should sell prospective customers on cost savings, while also delivering flexibility.
- More than 40% of organizations surveyed for this report do not have formal processes in place for comprehensive financial analysis of software commitments, even though most consider cost savings to be the primary motivator for open source adoption. This reinforces the value in including a calculator tool as part of this report for use in quantifying the financial benefits before and after project implementation.

TABLE OF CONTENTS

ABOUT THE 451 CAOS RESEARCH SERVICE

The 451 Commercial Adoption of Open Source (CAOS) Research Service focuses on the business realities of using open source products and practices within enterprises and software vendors. Its emphasis is on opportunities that open source methodologies offer, as well as the impact of consequent organizational, legal and cultural disruption. As part of the service, The 451 Group will publish six reports annually – each one examining a different issue and offering insight into user and vendor experiences. Additional features of the service include a weekly update of analysis and marketplace activity. The 451 Group also publishes a free blog for the enterprise open source community – 451 CAOS Theory – which can be found at: http://blogs.the451group.com/opensource/

Section 1: Overview

Open source software is rapidly gaining acceptance in the enterprise, and the reasons for this vary. With continued cost pressure on IT budgets even as organizations grow more dependent on IT to drive business ('doing more with less'), open source is an appealing option.

We've been told that achieving cost savings is a main reason for open source adoption, and this report is an exploration of this topic. We've verified this belief with our own survey results and interviews with IT end users. In our research, we've also found that formal IT processes are rarely in place for capturing and presenting the cost savings associated with open source adoption. As part of this report, we've created a vendor-neutral, impartial process for IT end users to calculate the potential cost savings of open source adoption.

The information presented in this report comes from the author's personal experiences in implementing open source technology in enterprise IT settings and as an open source consultant, as well as from interviews with open source end users and a number of excellent sources on the topic. The content of this report is not focused on a specific open source technology and is not vendor-specific. Therefore, the report and the associated calculator can be used to help identify potential financial benefits, whether considering the deployment of a single open source component in the datacenter or a desktop migration for an entire organization. The IT project under consideration may even involve moving from one open source offering to another, and thus there is a need to understand the financial impact of this move.

We provide an introduction to the basics of financial analysis – concepts, processes and elements – as they relate to the open source adoption process, and we provide a tool to help identify and capture the costs (and potential benefits) for the initiative in question. In addition, we support the analysis with examples and findings through our own end-user survey.

In our first 451 CAOS report, 'Stack and Deliver,' we examined the open source stack provider space and offered recommendations for these vendors and their customers. The material in our first CAOS report was useful for enterprise end users in gaining an understanding of the stack space, specifically in relation to making product and services buying decisions, mainly focused on the issues of open source support and component standardization efforts.

With this report, we step back and focus on one of the earliest phases in the open source adoption process – supporting the business case for adoption through the process of financial analysis. While not all open source adoption efforts are driven by cost savings, this is becoming a more common theme, especially when it relates to 'top down' initiatives driven by IT leadership.

This second CAOS report is a departure from the vendor-focused themes explored in the first report and is reflective of our commitment to cover the adoption of open source from many angles. Over the course of the next year and beyond, we plan to deliver analysis on a variety of topics relating to enterprise open source adoption, from the perspectives of software vendors, enterprise end users and the investment community.

Whether you are an end user with an internal IT project decision to make, a systems integrator considering an open source deliverable or an independent software vendor looking at

embedding open source components into your software offering, calculating the financial benefits of open source can help determine whether to pursue a project, discover how much it may cost to generate anticipated benefits and prepare for the potential risks involved. While this report is aimed at IT managers and architects tasked with building a business justification for open source adoption, we believe the material will be of value to software vendors that either complement open source technology or compete with it. We also believe this report will be of value to software vendors and investors that want to gain a deeper understanding of the process that IT end users must go through to proceed with a proposed open source initiative, and the overall role of cost savings in open source software adoption.

The only position this report takes is that financial benefits are possible with open source. The idea of cost savings is often synonymous with open source, but IT end users may lack the tools to effectively quantify the cost savings. Several research surveys, along with our own research, have shown that IT executives claim the main reason for selecting open source over proprietary software is either lower cost or avoiding vendor lock-in. Other common motivators include flexibility, security and performance. But the potential for cost savings continually rates as a top issue for open source adoption.

Section 2: Introduction

This 451 CAOS report, 'Cost Conscious,' is a practical guide for understanding and calculating the financial benefits of open source in enterprise IT projects. This report is meant to be actionable. It was written for the IT manager or architect, who, often with no background in accounting, is tasked with building a financial analysis for a proposed open source initiative.

The focus of this report is on the financial analysis of open source, not the business case for adoption. While the financial analysis supports the business case, it is important to understand that building a comprehensive plan for open source adoption is not within the scope of this report. The material in this report can be used to support the overall business case for adoption.

Before we provide the details of the calculation process, we feel it is important to provide an overview of the role of cost savings in the adoption of open source software, as well as the basics of financial analysis and the various cost elements involved.

2.1 THE ROLE OF COST SAVINGS

Gaining a firm grasp on the financial implications of open source adoption is crucial to any IT organization, but it's not an easy task. While the opportunity for financial benefits may seem like the predominant factor driving adoption, it is not the only factor. Considering the opportunity for financial benefits along with the flexibility that comes through access to the software's source code, the freedom from vendor lock-in, and the potential for improved security, reliability and performance, open source is a serious contender in enterprise IT today.

Of the dozens of organizations interviewed for this report, almost half said that financial benefits were the primary reason for adopting open source, and three out of four indicated that cost savings were an important factor in the decision-making process. For those who said that cost savings were not the primary reason for adoption, the responses tended to relate to greater control, whether that was the avoidance of vendor lock-in or the ease of customization that comes with access to the software's source code.

The key take-away from these surveys and interviews is that most end users are embracing open source software to save money, and they are gaining an increased level of vendor independence and improving flexibility of the software and how it's used across an organization.

2.2 THE 'FREE' IN FREE SOFTWARE

Prior to the use of the term 'open source' in the late 1990s, the IT world referred to this movement as 'free software.' The founder of the free software movement, Richard Stallman, is well known for his quote – "Free as in speech, not beer." In the English language, of course, the word free is used to mean both zero-cost and liberty. Hence, it is important for us to dispel a myth regarding the 'free' in free software. This report concerns itself with the zero-cost aspect of free software.

Free software seems compelling, but the reality is that the term can be misleading. While the software licenses may come at zero cost (and this is not always the case with open source), there is a cost associated with deploying and maintaining the software. Eric Raymond makes this point in his essay, 'The Magic Cauldron.' "The term 'free' is misleading in another way as well. Lowering the cost of a good tends to increase, rather than decrease, total investment in the people and infrastructure that sustain it." Meanwhile, Jamie Zawinski, formerly of Netscape Communications, is quoted as saying "[Linux] is only free if time has no value," while Scott McNealy, chairman of Sun Microsystems, quipped that "Open source is free like a puppy is free."

2.3 BUILDING THE BUSINESS CASE

Like any technology decision, the adoption of open source requires a business justification, and the issue of cost plays a significant role in this decision. Calculating the costs associated with a proposed open source implementation is an important step in building the business case for open source, but it should not be considered the sole criteria for making the business case. Rather, cost information must be placed into the context of the business' overall needs. There are benefits and risks with open source that are not financial in nature, and these need to be considered as well. See the 'Resources' section at the end of this report for a list of sources that can provide help with the building of a business case for open source. This report also does not attempt to cover the topic of selecting open source software; this topic will be explored in a future 451 CAOS Report.

Whether or not an open source deployment will result in financial benefits, it is valuable to calculate the costs of the deployment, even if the primary driver is not cost savings. Regardless of the organizational type, project scope or level of complexity, and whatever the reasons are for considering open source, there is substantial benefit in understanding the financial impact of this decision.

This report is intended to provide value to IT end users at any phase of the open source adoption process. The research and analysis in this report are relevant whether an organization is in the evaluation phase of a migration, is preparing for an implementation of a new initiative or needs to better understand the finances of an existing open source deployment. The information in this report is applicable to an international audience – all references to US dollars or any specific currency have been left out of the report and calculator with the explicit purpose that the report speaks to the business requirements of readers worldwide.

Section 3: Financial analysis 101

Before we go any further, it's important to review the basics of financial analysis. For some, this will be new information, and for others, this section can be skimmed over. Let's assume that you have been asked to provide a financial analysis for a proposed open source initiative, and a manager wants the analysis by the end of the week. Now what do you do without a background in accounting? It's one thing to talk about the financial benefits of open source, but how would you go about calculating them? How do you account for the unique aspects of an initiative and an organization? What sources can you trust? What costs do you need to capture? What calculation method(s) should you use? How should you present the data? These are exactly the types of questions we will address as we proceed with this report.

This section is designed to provide an education on the various terms and models used in capturing and calculating costs (and financial benefits). You may have been tasked with building a financial justification for an open source initiative, with little or no knowledge of how to effectively capture costs. Since each organization has unique internal processes and requirements, this section covers the topic in a practical manner, applying details specific to their circumstances. In addition to covering specific terms, this section will include comparisons of ROI and TCO, soft and hard costs, capital and expense budgets, recurring and onetime expenses, and other financial terms.

It's important to understand that one does not need to have a background in finance or accounting to do this kind of analysis. It's also important to realize that there are several different styles of building a financial analysis, and that these styles vary from organization to organization. What's more, there are no standard ways for presenting this information.

There are many ways to estimate cost savings for an IT initiative. Among the most popular metrics used, based on industry surveys (including our own) and discussions with end users, are net present value, total cost of ownership and the annual cost savings of a project. We will explore these and other metrics in this report.

3.1 THE VALUE OF FINANCIAL ANALYSIS

Understanding the financial impact of a decision is important and helps an organization move forward in its decision-making process. When approached properly, financial analysis can become an objective process for helping make decisions and can be used in various business units within an organization. It accomplishes three central objectives: first, it provides greater confidence in the cost considerations of the project; second, it provides some information about the overall impact of the project on the IT budget; and finally, it supplies the senior management team with the basic information to perform a rigorous cost/benefit analysis, especially when comparing multiple IT projects.

In business, when it comes to the allocation of resources, the costs and benefits of a project play a critical role in the approval process. While this report does not cover all the benefits of open source, it certainly addresses the financial issues. The relative importance of cost in the decision-making process is based, to a degree, on the financial health of an organization. In

general, IT budgets remain under cost pressure. Detailed financial analysis will ensure that IT leadership has the supporting data to back the proposed initiative.

IT budgets have risen only slightly over the past few years, based on industry data, and they are growing at a rate slower than overall business growth – meaning that IT budgets are essentially getting smaller as a percentage of the overall business. The important question is how this generally slow growth in IT budgets impacts the overall growth of businesses, and whether that will figure into businesses' financial analysis of an open source initiative. In general, when revenues are rising, businesses are doing more, but in the case of IT budgets today, they're doing more with less.

This all points to the fact that the cost of a project remains an important factor in the approval process (as well as a project's ability to generate revenue). Financial analysis can be used by executives to compare the benefits of multiple initiatives competing for the same limited resources. Financial benefits can be a significant motivator, even when other benefits may exist. For several of the organizations interviewed for this report, financial benefits were the primary reason for convincing upper management to approve the proposed project.

Unique projects, unique organizations, diverse financial elements and rare formal processes (just to name a few issues) all point to the reality that financial analysis is difficult. This process is much more of an art than a science.

In the end, the process of generating a financial analysis is not just about proving the financial benefits of a given project. It forms the foundation of understanding for subsequent projects and can provide a clear impact statement regarding the integration and development of new technology so this technology can be compared with other projects. A financial analysis is a single source of information, although an important one. It indicates how much it may cost to generate anticipated benefits, and it helps to prepare for any potential risks that are involved.

3.2 TEN CONSIDERATIONS

We have outlined the case for financial analysis, and how this relates to open source software. Here are 10 considerations to keep in mind with regard to crafting the analysis:

1: Not every project results in a financial benefit.

Not every IT project is pursued based on financial reasons alone. There are other factors here, including strategic value or the elimination of legacy systems, for example. In addition, not every project that anticipates a financial benefit actually generates one. Things change, estimates are proved incorrect, and what looked like an airtight financial analysis may have failed to account for an important detail in the end. Even though the opportunity for cost savings is one of the top reasons for an organization to pursue an open source initiative, cost savings are not always the result. Open source may not be the lowest-cost option available to an organization.

2: Each project is unique.

As we mentioned above, not every project is approved based solely on its financial benefits. Other important factors are at play, including the generation of revenue, the support of other projects, strategic value and the overall level of confidence. It would be very convenient to have a 'one size fits all' approach to financial analysis, but this isn't possible. Each project is unique, and this uniqueness must be captured and quantified. Not all projects have the same financial elements, with software licenses, hardware and training being several examples of elements that vary by project.

3: Each organization is unique.

Organizations have their own processes, and sometimes these include standards, or even government regulations specific to an industry. As such, industry-specific or organizational-specific aspects must be considered. As Neil McAllister indicates in his article entitled 'You can't kill TCO,' the unique combination of resources, both machine and human, at work within an organization is something that can only be fully understood in the individual context.

4: Formal processes are not always in place.

Not every organization has formal processes in place for financial analysis. A significant number of the organizations interviewed for this report and those participating in the survey did not have such processes in place. Without a formal process, it is difficult to effectively compare projects and track their true financial impact over time. This report does include a process, and if an organization already has a process in place, this report can be considered as offering an opportunity to augment it, not replace it.

5: Financial analysis is often a comparison.

When preparing a financial analysis, it will likely compare two or more alternatives. This could be an existing setup compared with a proposed project, or even multiple proposed projects. It may be a comparison between a proprietary offering and an open source alternative, or even the possibility of 'build vs. buy.' The decision may also relate to an end-of-life software installation, with the desire to compare the proposed replacement with what existed before. It's important to remember that financial analysis is most valuable when there are multiple options, and the pros and cons can be viewed across multiple scenarios equally.

6: Consistency is key.

When comparing multiple options, accuracy and consistency with the calculations are key. The most important point to remember is that the same standards should be applied in any comparison. Treat each option equally. This is the only fair way to compare projects.

7: Cost savings are rarely 'saved.'

In the cases where there are cost savings, what becomes of the savings? Of the organizations interviewed, the vast majority said that the savings were reallocated to other IT projects. Realizing savings rarely means that the money is returned to the organizational coffers or

the shareholders. By saving money in one project, the money can be freed up to be spent in another area. These savings often create new opportunities, instead of actually being 'saved.' Also, this may never actually equate to a reduction of expenses at the departmental level the next year. In the case of an ISV or systems integrator, cost savings can be the impetus for cheaper offerings for the end customer, or increased margins.

8: Financial benefits studies often support a position.

Many software vendors – proprietary and open source alike – provide marketing materials that praise the financial benefits of using their software. There is a plethora of studies attempting to show the cost benefits of one product over another. In almost all cases, studies that are sponsored by vendors show a positive return, as might be expected. Some independent studies do exist (such as this report), although it is not often clear how independent they really are. We make it clear that the only position that this report takes is that financial benefits are possible with open source, but are not always the case. Maria Winslow covers what to look for in a study in her book, 'The Practical Manager's Guide to Open Source.' While the book may offer good reference points, they are not fully applicable to the unique aspects of both a project and an organization. Review them critically, as you would this report.

9: IT budgets aren't just about IT.

Although CIOs are generally the final decision-makers for IT budgets (in organizations large enough to have a CIO), some organizations have other structures in place for financial decision-making. Many organizations place business leaders and the CFO in the critical path for project approval, and it's important to 'speak their language' to effectively gain approval to pursue the proposed project. This fact is enough to warrant a serious financial analysis, since this information may be requested beyond the IT organization.

10: Financial analysis is not the same thing as a budget.

Although financial analyses and budgets both involve the calculation of costs, they serve different purposes. A financial analysis for an open source initiative is focused on the justification itself and is not an operational approach. A budget, on the other hand, is used for the management of the project rollout. It's important to understand this, since the focus of this report is on offering aid not in creating a budget, but in understanding the financial benefits of an open source initiative. Much of what is created in this process can be applied to a budget, if the project is approved.

With these considerations in mind, let's explore some of the important terms that will come into play in proceeding with the financial analysis.

3.3 GLOSSARY OF TERMS

The financial terms described below are frequently cited by the organizations we cover, and they play a crucial role in how the calculator is used in the financial analysis. While the application of these concepts may vary from one company to the next, their meaning is generally understood by business professionals. Don't get bogged down by these terms, but it's important to understand them as we move forward. The first four terms refer specifically to

metrics commonly used in calculating financial return, followed by supporting terms that will be used along with these metrics.

Total cost of ownership (TCO)

The total cost of ownership is the total cost of an item or project over time. TCO is considered a primary metric used in calculating financial benefits. The term can be used in the context of 'lower TCO' or a 'reduction in TCO,' which typically relates to an existing system. The results of a TCO calculation are often defined in a time period, such as the TCO per year or the TCO over a product's useful life. When purchasing a server, for instance, the TCO of the server would be calculated, including the hardware costs along with other associated costs, such as support, extended warranties, installation, maintenance and operations. The TCO can provide insight about a project's relative financial benefits compared with other options. The actual TCO calculation requires several other financial elements, which are explained in further detail later in this report. Regarding the 'ownership' aspect of TCO, the licensing terms of open source software may be such that the software itself isn't actually owned as we think of proprietary software licenses being owned, but the ownership we are concerned with here is the entire project and its related costs.

Return on investment (ROI)

The return on investment is the profit or loss associated with the investment over time. ROI can be used to determine the annual rate of return, or the payback period to recover the original investment. The ROI is often expressed as a formula: (Return – Investment) / Investment. If something costs 1,000 and it generates a 3,000 return, then the ROI would be 200%, which is to say that it generated 200% more than the initial investment. While this is a very simple formula, the data inserted into this formula may not be easy to determine. If it takes six months to generate 1,000 from the investment, then the payback period is six months. In that six-month period, it has reached the break-even point – the investment money has been recovered. If the investment generates 1,500 per year, then the annual rate of return is 1,500, or expressed as a percentage, 50%.

Net present value (NPV)

The net present value is the value of money in the future compared to the value of money today. This concept is often referred to as 'the time value of money.' That's because money is worth more today than the same amount of money in a year, because one can invest that sum of money today and have more money a year from now based on that investment. The value in determining NPV is that it provides a way to compare projects that have different time horizons. With NPV, a project with a three-year life can be compared financially to a different project with a five-year life. Companies generally use a firm-specific interest rate, commonly known as the cost of capital, in discounting future cash flows to arrive at today's value. The cost of capital is expressed as a percentage, representing the cost of financing an organization's operations. An organization is aware of its cost of capital; the calculator in this report has a built-in figure of 12%, which may need updating based on an organization's own cost of capital rate. In theory, should the NPV be greater than zero, the project will make money. The larger the NPV, expressed in money, the more compelling the project opportunity is. When the NPV

is positive, then the project will generate returns higher than the cost of capital used to make the investment. Using NPV to determine the financial value of a project is a form of ROI.

Internal rate of return (IRR)

The internal rate of return is the interest rate that would make future investments result in no payback, based on the value of investments. It is an approximate number derived through trial and error. If the IRR value is negative for a project, then the money would be better off invested elsewhere. NPV and IRR are inherently related, despite IRR being expressed as a percentage. The IRR is often referred to as a 'hurdle rate' – a decision factor in whether the investment will make a significant return versus other investment opportunities. When IRR exceeds a company's own cost of capital, the project is financially justifiable. It is a common way to compare investment opportunities, including other IT projects competing for the same budget allocation. Using IRR to determine the financial value of a project is a form of ROI.

Cash flow

Cash flow is the amount of money that is being spent or gained by an organization during a period of time (amount gained – amount spent). If during a specific year, the project spent 10,000 and generated 15,000 in revenue, then the cash flow for the project for that year would be 5,000.

Hard and soft costs

Hard ('tangible') and soft ('intangible') costs are, respectively, expenses that can be easily quantified and those that are more abstract and difficult to quantify. One can easily determine the cost of a server from the invoice, but calculating the cost of evaluating a product or service (time spent) or the benefit from improved productivity (time saved) is much more difficult. Organizations treat soft costs differently, and some are not allowed to quantify soft costs at all in the financial analysis.

Recurring and onetime costs

Recurring and onetime costs are costs categorized by their frequency. A server purchase may be a onetime cost, while a support contract for the server is recurring. In effect, this means the support costs continue over a period of time. Some of the calculated costs will be onetime costs, while others will be recurring costs.

Up-front costs

Up-front costs are costs at the beginning of a project – the initial investment made to start a given project. They are otherwise known as the cost of acquisition (COA), switching costs or the outlay of initial capital. These can be considered 'switching costs,' when moving from one product or service to another. For the calculator, we consider these costs as the initial investment, prior to deployment.

Opportunity costs

Opportunity costs are the costs (or loss of potential revenue) incurred by not pursuing an alternative course of action. By choosing to pursue this project over another, what losses were incurred? Would the alternative course of action have generated more revenue?

Cost avoidance

Cost avoidance is the flip side of opportunity costs. By taking this course of action, how much money was saved by avoiding future costs associated with the other course of action? Cost avoidance should not be overlooked, especially with regard to the pricing of software maintenance contracts.

Sunk costs

Sunk costs are money that has already been invested and cannot be recovered. This is especially important when comparing an existing implementation to a replacement project. The amount of sunk costs in an existing implementation may be a barrier to spending new funds on a replacement, in some cases. As Paul Kavanagh says in his book, 'Open Source Software: Implementation and Management,' "At this time, switching and sunk costs and de facto industry standards are tending to delay adoption of open source, particularly at the desktop." However true that statement may be, an investment opportunity that has the ability to generate a compelling return greater than the sunk costs in a reasonable amount of time should be pursued.

Fixed, variable and incremental costs

Fixed, variable and incremental costs are costs categorized by their nature. Does a specific cost remain the same (fixed) as the organization grows, or is it variable, depending on the growth of the organization? Or is the cost incremental, such that it occurs over time based on business activity?

Loaded costs

Loaded costs are the total costs for a specific financial element of a project. The cost of labor is a good example of a loaded cost. When calculating the loaded cost of labor, the total includes not just the salary amount, but also benefits, vacation time and possibly facilities costs. You can think of a loaded cost as the TCO focused on one specific cost element.

Capital and expense budgets

Many organizations have separate budget allocations for capital and expenses. Capital, often referred to as 'Capex,' is money used for investment by an organization. Capitalized funds are often treated differently for tax purposes and depreciation. Software development and the purchase of goods, such as hardware, are often considered capitalized, while subscriptions to services and the cost of doing business are not. These are important terms to understand, since the money available for a project may already exist within capital and expense funds.

Depreciation

Depreciation is the decrease in value of an asset over time. Depreciation is often calculated by an organization for capital expenditures, such as servers. A server that is two years old does not have the same market value as a new server, for instance. Depreciation is important to understand, since the duration of the financial analysis will impact the depreciation requirements for some of the financial elements. Not every organization is required to figure depreciation into a financial analysis. An accounting department often does this after the money has already been spent.

Amortization

Amortization is the distribution of costs into smaller payments over a period of time. Sounds like incremental costs, doesn't it? Amortization can be considered a type of incremental cost. The cost is incurred to an organization through a single payment, but for internal bookkeeping, these payments are charged over a period of time. Not every organization is required to figure amortization into a financial analysis. An accounting department often does this after the money has already been spent.

3.4 SELECTING A METHOD

We have defined various methods for estimating the financial impact of a proposed project. Among the common metrics used in financial analysis are total cost of ownership (TCO), return on investment (ROI), internal rate of return (IRR) and net present value (NPV). The decision about which metrics to use should be based on the organization itself and what kind of analysis needs to be delivered.

These four metrics are among the most widely used tools in calculating financial return. There are other methods of calculation, but it would be impractical to cover them all in this report, since they are often industry-specific. Each of these four common metrics has its own value. When using a combination of them, one can build a much clearer picture of the total financial impact of a proposed project.

The calculator will incorporate these four metrics, which should cover the majority of needs. Ideally, the financial goal of a project is to reduce the TCO, increase the ROI (understanding the return rate and the payback period), and generate a positive NPV and IRR.

In preparation for using the calculator, it's important to understand the various financial elements that need to be quantified within the financial analysis.

Section 4: Elements of the financial model

The preceding section provided an overview of the rationale, metrics and terms that are fundamental in the process of building a financial case for open source adoption. In this section, a distinction will be made between hard costs – financial costs and related benefits that can be easily quantified – and soft costs and related benefits, such as developer productivity and manageability, which are harder to calculate and are often not accepted by organizations as part of a financial analysis.

To calculate the potential financial impact of an open source initiative, there is a need to understand the various costs and possible benefits at play – referred to in this report as 'financial elements.' The financial elements listed in this report are types of costs that are commonly encountered in an IT project, as well as some related benefits. This is by no means a complete list, and it is important to account for financial elements not covered in this report by also understanding organizational-specific requirements. The main purpose in reviewing the various financial elements is to have an awareness of a broad range of costs to be considered.

Another issue to keep in mind is that of 'hidden costs,' or costs that are not always clear up front when doing the analysis. Surveys have indicated that nearly half of IT end users typically encounter significant hidden costs in open source initiatives.

Some of the financial elements are hard costs, while others are soft costs. As we've said, many organizations do not support the quantification of soft costs when building a financial analysis, so it's important to split the financial elements into different categories. These categories are not limited to just hard and soft costs, however, as we cover internal costs, evaluation costs and even revenue; these elements may or may not have an impact based on an organization's particular requirements.

We cover the various financial elements before introducing the calculator because this is crucial information for doing a thorough financial analysis. It is essential to understand and identify costs before getting started with the calculator. It's important to keep in mind that the more detailed the financial analysis is, the more accurate it's likely to be. Choosing to disregard a known cost in the calculation process leads to an inaccurate view of the financial reality.

The other important point to grasp here is that some costs aren't relevant. Each project is unique, and the specific financial elements will need to be captured. When comparing a proprietary offering with an open source offering, the costs structures may be different, and this may even be the case when comparing multiple open source offerings. Review the list of cost elements in this section, and look for those elements that will apply.

Not every calculation is easy, especially when it comes to soft costs, where a fair amount of estimation will need to occur. We will cover this in more detail in the section on soft costs. This section should be reviewed again when working with the calculator.

4.1 HARD COSTS

The hard-cost financial elements listed below are the ones that are both quantifiable and most likely to come up in a proposed IT project. Each of the following hard costs is part of the foundation on which to build a financial case for open source adoption. Here we cover the 10 most common hard costs, included in the calculator, and list additional hard costs for consideration as well.

4.1.1 Software licenses

Software licenses are one of the most apparent areas for open source cost savings, either for a new IT project under consideration or for an existing IT project that anticipates growth. When referring to software licenses in this report, we are talking about the costs of the licenses themselves, not the legal licensing terms or conditions. Don't confuse this financial element with the legal agreements in place for using open source software, often referred to as 'open source licenses.'

The potential savings from software licenses may be the most dramatic source of financial benefit for an open source initiative. Then again, it may be a nonissue. Cost savings on software licenses are seen as the greatest opportunity for savings, even though this cost is often only a fraction of the total cost of an IT project over its lifetime. When considering an open source initiative as a replacement for an existing IT project, the cost savings related to software licenses will likely be much lower than the savings for an entirely new IT project, since money will already have been spent in this area. Although, if upgrades from an existing proprietary system are planned, this will likely be an issue.

Zero-cost licensing for open source software can make system growth more cost-effective. Growing a deployment with additional servers requires no additional fees to be paid for the software itself. The savings from software license fees in an open source initiative have often been used to offset added costs, such as training and professional services.

Open source software is not universally zero-cost software, however. A number of open source vendors offer a dual model, where a version of the software is free for use, and another version – one with added features or one approved for commercial reuse – requires a paid license. There are also some vendors that charge for their open source software, often through subscriptions. If the open source software intended for use has a license cost associated with it, then it will need to be treated essentially the same as proprietary software when calculating the costs of software licenses.

When looking at a system, it may include multiple software components that need to be calculated, and some may have a software license cost. All of the software licenses required for a particular deployment and in support of this deployment need to be captured and calculated. These could be server software licenses, client access licenses or desktop licenses, for example, or even related software costs, such as database licenses or development tools.

Often, although not always, software licenses are priced based on the number of CPUs the software is running on. Some software licenses are based on the number of users. The pricing can vary based on whether the software is being used in a production or development environment. With the emergence of multicore CPUs, virtualization, and utility and grid

computing, the pricing of software licenses has become more vendor-specific, with each vendor essentially using its own rules. This may make the process of calculating these license costs more difficult.

Assuming that the selected open source software components have no software license fees, then the cost in this category will be zero. When comparing two open source offerings, or when comparing an open source offering to an existing proprietary one, the software licensing cost is zero for all options (since the cost of the existing proprietary offering is a sunk cost anyway), and this is a nonissue. However, when comparing a proprietary offering with an open source one for a new project, the difference can be dramatic. To show the potential savings from choosing the open source option, a comparison would have to be made with an option that includes software licenses fees. Otherwise, the cost is simply zero, and there is no actual calculated benefit to an organization from choosing the open source option. Making this comparison is how 'cost avoidance' is determined for an open source initiative.

Not all software licenses represent onetime costs. Software licenses may be in the form of a subscription, or they may be volume or site licenses. This can change the method of calculation. Software licenses may also be incremental in nature, with costs occurring over time as the deployment grows. Estimating these future software license costs will help in estimating future financial benefits related to software licenses. The same holds true for known future upgrade costs associated with software licenses. One must estimate all of the software license costs over the relevant time period, assuming that the zero-cost open source option is not selected.

Software licenses aren't always priced the same over time. Licenses may become cheaper or more expensive in the future, and these costs must be estimated when calculating total cost over a multiyear period. Recently there has been some indication that zero-cost open source software has caused some proprietary software vendors to lower their software licensing costs. This trend is likely to continue as open source software becomes more pervasive.

The costs associated with software licenses are best captured through historical invoices, and by talking with software vendors directly. The use of list pricing is not always accurate. What an organization paid or will pay for software does not always match the sticker price. The other issue to be aware of here is that some software licenses, such as operating system licenses, are attached to a hardware invoice.

4.1.2 Hardware

A new project may require new hardware. When considering the replacement of an existing system with an open source one, can the same hardware be used? There are potential risks (e.g., rollbacks and downtime) in using the same hardware, and there are also potential financial benefits in using the same hardware. There may be a preference to purchase additional hardware for a project. If thorough testing of a new system needs to be done prior to rollout, while continuing to support an existing system in the interim, new hardware may be necessary. Can the current production system be taken down for the migration? Hardware reuse is an important issue to consider.

Open source software sometimes has reduced hardware requirements compared with a proprietary alternative, or it can make better use of the hardware than proprietary software.

This needs to be determined on a case-by-case basis, but in the author's personal experience, money has been saved on future hardware purchases by making better use of the existing hardware through open source software.

Will the proposed open source initiative impact desktop users? If so, the hardware costs for the desktop associated with this proposal must be factored in. Will this impact any other hardware costs, such as networking equipment, appliances or development workstations? And don't forget to capture any hardware maintenance costs, although this is discussed in a different section of the report (see 'Maintenance').

The effective lifespan of the hardware will need to be determined. Will the existing hardware last for the duration of the period covered in the financial analysis, or will it need to be replaced? Hardware upgrades need to be factored into the calculation as well.

4.1.3 Support

When considering the support of open source software, there are often more options than with proprietary software, although they are not always as mature as proprietary software support options. Support for proprietary software is generally provided by a single vendor and, as the sole provider, the vendor sets the pricing. With open source, the opportunities for support depend, to a large extent, on the relative popularity of specific open source software components. The more popular open source software components typically have a greater range of support options available from a number of sources.

The level of support required is best understood within the context of the type of deployment under consideration. A production environment and a development environment have different support needs. The more critical the system, the more critical the need for support is. The number of support incidents, hours of operation and named contacts are all important aspects of comparing support options.

The three most common models of support for open source software are: professional support by open source software vendors, third-party vendor or consultant support, and self-support. Professional support is offered by the primary caretakers of the open source software in question, and the support offerings often resemble proprietary support offerings delivered via a support contract. Not all open source software components are backed by professional support. Third-party vendor or consultant support is offered by vendors or individuals with expertise in the open source software in question, and these offerings vary. Self-support (also referred to as community support) brings the support responsibilities in-house, with additional support provided through interaction with the open source community. Each of these options has its own costs, benefits and risks.

There are several risks associated with both third-party and self-support that are important to understand. In order to assess these risks, one needs to ask several questions when deploying open source software in a production environment. Most important, does one want to be dependent on the resources of the open source community when the system is down and an organization is losing revenue?

Incurred downtime does have bottom-line impact, potentially in lost revenue and labor. Service-level agreements may not be offered by third-party providers, and they are certainly

not offered with community support. In addition, if indemnification is an issue for an organization, this is only obtainable through a professional support option from an open source software vendor for their particular supported software. An open source project team with no legal structure cannot provide this, and neither can third-party providers. Self-support provides the opportunity for more internal control, but at a greater risk.

Self-support, on the surface, looks to be the lowest-cost alternative. But this approach may not result in the level of support needed for an organization. Based on the level of need and the available providers, open source software support may not be that much cheaper than proprietary software support. Open source companies that do not charge for software licenses are dependent on support revenue, while proprietary software vendors have a greater opportunity for overall discounting, since they are collecting software-licensing revenue as well.

When considering multiple support options for an open source initiative, these should be captured separately and presented as multiple options within the calculator.

When pursuing a professional or third-party support arrangement, there is a benefit in having this support agreement in place prior to deployment. This can be a helpful way to better understand the open source software during the development and testing phases of a project. However, the down side of this is that an organization could end up paying for support too far in advance of when it's needed most, in production. There is a benefit to buying support a bit early, but not too early, or you'll be paying for support you won't use as much.

If an initiative contains multiple open source software components, it may not be possible to obtain support for all of them from the same source. A new category of open source support providers called 'stack providers' has recently emerged, and these companies were examined in our first 451 CAOS report, 'Stack and Deliver.' In most instances, it will be necessary to build a custom open source support plan for an organization based on the various offerings and providers used. This can make the calculation of open source support a difficult process, since it includes some soft costs (e.g., internal labor).

For a more detailed overview of open source support options, please refer to 451 CAOS Report 1, 'Stack and Deliver.'

4.1.4 Development

Support issues are a critical underpinning in any open source initiative, but also important is the issue of software development costs. It can be argued that access to the source code makes development easier and less costly, but any new IT project has the potential for new development costs.

Migrating to a new open source system may also be an opportunity to do performance tuning, and a bit of re-coding may be required for software to work properly with the new system. What's more, an open source software component may not have feature parity with a proprietary component, thus requiring some custom development.

All of these factors must be considered. However, it's important to approach development as an opportunity to capture the costs for near-term enhancements, based on business requirements. That's because it may be necessary to include feature enhancements with the open source

option to match the features of a proprietary option. If the total development costs for the new deployment are known, then include these costs in all scenarios.

The issue to be aware of here is that adding custom code into the open source component may make it difficult to support in the future. Keep in mind that customizations for a particular business may need to be continually re-integrated into future open source software releases. If there is functionality required from an open source component that is not part of its core functionality, consider working with the open source project team on meeting this functional need, possibly even contributing the resulting code back to the project.

Regarding the issue of internal labor as it relates to development costs, some organizations do not account for this, since it is already accounted for as overhead costs for the organization, regardless of the specific IT project that is being pursued by internal developers.

4.1.5 Professional services

Professional services are often categorized as related to development, installation and configuration. It's important to treat professional services as a separate financial element for the calculator because there needs to be a clear distinction between the core development requirements of an IT project and any supporting professional services, such as those provided by a software vendor.

If the development of the open source initiative is being outsourced, then these costs should be treated as development costs, not as professional services. Consider this category to be for additional external help – either from an open source vendor or a third-party vendor, or an individual. Do not include training in this category.

4.1.6 Training

When dealing with a new system or technology, training is an important aspect of educating the project team. A better-informed team can better address any issues that arise during the project. Training comes in many forms, and it may be provided directly by a software vendor or through a third party such as a professional learning center or an educational institution, and it may occur on-site, off-site or even online. It may even be appropriate for the training to be done internally by and for the team. If any training costs are associated with a proposed project, these need to be captured.

Developers are not the only beneficiaries of open source training. Operations staff, such as systems administrators and even quality-assurance staff, may benefit from this training. When deploying a desktop system or any system that impacts end users, retraining is often required. Consider the potential benefits from training and devise a training plan specific to an organization's needs.

4.1.7 Testing

Systems need to be tested before they're deployed, and it's important to capture the costs associated with testing. Whether involving unit testing, performance testing, functional testing, creating test scripts, the documentation of use-case scenarios or other forms of testing, quality-

assurance costs must be factored into the financial analysis. The testing needs for a migration are different from those for a new initiative, and they should be treated differently. Work with the testing team to determine these costs.

4.1.8 Operations

What will it take to operate the open source deployment once it has been deployed? This issue is often referred to as 'manageability.' Operations costs may be a mix of labor (systems administrators), management and monitoring tools configuration, and even the creation of manual procedures to support operations. Open source deployments tend to have less mature management capabilities. Custom development to support operations should be treated as a development cost, and any software licenses for monitoring tools should be captured as software license costs.

With open source software, the potential for initial operational challenges and the potential for improved reliability may balance each other out. Work with the operations team to determine these costs.

4.1.9 Staffing

Will the proposed open source initiative require any additional staff to develop, support, test or operate it? It is valuable to capture these staffing costs independently from the other financial elements, since hiring is treated uniquely in many organizations. If an organization is hiring for a specific open source skill set that it currently doesn't have in-house, it should carefully review the going rate for these skills and consider this information when estimating the cost of staffing. Some studies indicate that open source talent is less expensive, while other studies indicate just the opposite.

4.1.10 Maintenance contracts

Maintenance contract costs for software and hardware need to be captured in the financial analysis. Maintenance fees from proprietary software and hardware vendors are typically based on 15-25% of the cost of the software licenses or the equipment per year. These maintenance costs are different from the internal costs to maintain software. Maintenance contracts are often confused with software subscriptions, which are a form of software licensing fees. Maintenance costs are typically calculated using the list price, and not what was actually paid.

While many proprietary software vendors treat support and maintenance as a combined cost, this is rarely the case for open source software. Software maintenance is tied to the maintenance of the software licenses themselves – supporting the costs associated with patches, updates and new versions. With most open source software being zero-cost, maintenance costs associated with open source software are rare, even with open source software vendors. Often the vendor will include maintenance with any associated subscription fees.

Hardware maintenance is also an issue, and it needs to be addressed. Vendors of most hardware devices, from servers to firewalls to routers, offer maintenance contracts of various levels. A move from a proprietary hardware platform to a commodity platform can result in substantial savings in this area. Make sure to capture these costs for the calculator.

This category may need to be treated as two categories in the calculation process – software maintenance and hardware maintenance – based on the complexity of the environment.

4.1.11 Other hard costs

The 10 most common hard costs we've just listed cover the majority of cost elements for most IT projects. But these are by no means the extent of the hard costs for a project. We have listed some additional hard costs below for consideration. An organization may have their own to calculate as well.

Migration: When migrating from an existing system, data may need to be moved to the new system, and it may need to be integrated with other internal or external systems. If the proposed project has a data migration component to it, these costs must be calculated. Data migration includes development and testing components.

Environmental: These costs include many of the costs associated with datacenters or hosting, such as datacenter floor space, power, bandwidth, shared infrastructure services or hardware leasing.

Documentation: These costs are often captured as part of the training category. Does documentation need to be created for the initiative?

Deployment: These are costs associated specifically with the deployment itself. This is especially important if some of the operations have been outsourced.

Configuration: These are costs associated specifically with configuration. This is often captured as part of the development and operations categories, unless the configuration needs are highly complex.

4.2 SOFT OR INTANGIBLE COSTS

Not every element in a financial analysis can be easily determined. Soft costs, also called intangible costs, can be difficult to calculate and are not always accepted as part of a financial analysis. Examples of soft costs and benefits include employee productivity and system downtime avoidance. While we have seen no definitive survey results on the applicability of soft costs among organizations doing financial analysis, our experience is that soft costs are rarely accepted in such analysis. Financial benefits from a reduction in soft costs can often be the result of a project, even if these costs were not captured initially. And while there is a risk in calculating soft costs, the benefits of reducing soft costs can be undervalued by an organization. Some soft costs may be treated as internal costs (see below) within an organization.

Thus, individual soft costs are not included in the cost calculator by default, although soft costs are listed as a row in the calculator. If an organization accepts soft costs and has the ability to estimate the amounts, then this element should be added to the calculator. The soft costs can be captured, depending on the organizational requirements, as long as they are treated fairly and the level of confidence in the estimate of soft costs is addressed. Organizations should consider adding a margin of error for any calculated soft costs to reflect the intangible nature of these costs.

When unable to obtain financial benefits from reducing the 10 common hard costs listed above, organizations should be wary of trying to find the financial gains they seek through soft costs and benefits. If an organization is spending money with the expectation that the return of intangible benefits such as productivity and reliability gains will result in an overall benefits payback, it should proceed with caution. That said, the book 'Mythical Man Month' by Frederick Brooks makes the point that "betting real up-front money for the sake of projected but iffy benefits later is what investors do every day." This is an individual call, and the potential soft benefits should be understood, even if they are not calculated.

Some examples of soft costs and benefits include:

Downtime – the financial impact of system outage

IP risk – legal costs related to litigation

License auditing risk – resources required to perform a vendor-required license audit

License management – resources required to manage the deployment of licenses and the purchase of additional licenses as the deployment grows

License negotiation overhead – legal costs and resources required in negotiating the software licensing contract

Planning – resources for project planning and overhead

Process inefficiencies – lost time and costs related to any process activities

Procurement overhead – purchase costs and resources required to procure the software

Productivity – efficiencies from the use of software

Reliability – the financial impact of improved system reliability and uptime

Support quality – resources required during the software support process.

The descriptions above may also apply in the inverse. What was a cost in the initial or proprietary option may now be a cost savings opportunity with the open source alternative.

4.3 INTERNAL COSTS

Calculating internal costs, or those elements that are specific to an individual organization, is an attempt to capture the costs that are generally above the project level, in order to provide a realistic look at the total cost incurred by a project. Many organizations have their own standards for calculating these specific financial elements. What are the organizational rules for accounting for internal labor? What about charge-backs to other departments? An organization's expectation of internal costs and how they should be calculated must be considered.

Some examples of internal costs include:

- Amortization
- Capital vs. expense budget allocations
- Charge-backs
- Contingency
- Depreciation
- Inflation

- Interest
- Loaded costs
- Management overhead
- Project overhead
- Taxes
- Time and internal labor

Individual internal costs are not included in the cost calculator by default, although internal costs are listed as a row in the calculator. An organization should add any of its required internal costs.

4.4 REVENUE

While cost savings may be the primary means of bringing financial value to an organization for an IT project, there may, in fact, be a revenue benefit in the mix. The move to a new technology may result in improved system performance, which may result in new revenue opportunities, regardless of whether the technology is open source or proprietary in nature.

While certainly not a 'cost,' revenue can have a profound impact on whether a project is approved. The author was previously involved in a customer-facing open source deployment where there was a clear correlation between the project and an increase in revenue, through increased system performance. The author did not account for this revenue gain as part of the initial financial analysis, since this was not an expected outcome. The savings in hard costs justified the project without factoring in any potential revenue gain. On the flip side, a proposed project may result in a revenue decrease. If this is known or can be estimated, this should be captured.

Revenue is included in the cost calculator row. It will need to be determined whether the potential revenue gains from a project can be accurately estimated. The book 'The Art of Project Management' by Scott Berkun effectively sums up this issue as: "deciding when to pay the tangible short-term costs for less tangible long-term revenues." This is a tough call to make. Consider adding a margin of error for any calculated revenue gains to reflect their intangible nature.

4.5 EVALUATION COSTS

Some organizations include the costs associated with the evaluation of technology as part of the overall cost of the project. We have chosen not to cover this as part of the calculator. We consider this as a phase prior to financial analysis and independent of the costs directly associated with the development, testing, deployment and operations of an IT project. This report is not intended to be used for the evaluation and selection of open source software. This is a topic we intend to cover in a future 451 CAOS report. To capture the evaluation costs, or to learn more about this topic, an excellent source is the Woods and Guliani book 'Open Source for the Enterprise.'

With the knowledge of financial elements in hand, it's time to begin working with the calculator.

Section 5: The calculator

The 'Calculator' section of this report is an introduction to the associated spreadsheet and the process of capturing the various costs and potential financial benefits of each individual financial element. Various uses of the calculator will be covered, including the process of comparing multiple technology options and determining the payback period and the costs of operation over a period of time. The calculator is a framework that can be used in a manner appropriate to the situation, as opposed to following a 'one size fits all' format.

It is now time to build a financial analysis using the calculator. The 'Financial elements' section was a review of some of the individual elements that play a role in the calculation process, and now it's time to pull this information together and calculate the proposed project's overall financials. This is financial analysis in action. Example uses of the calculator are included in this report.

The calculator is a simple spreadsheet that factors the costs and savings of technology projects over time. It is a multiyear financial analysis with side-by-side financial comparison of two options – one open source and one alternative, which may be an existing deployment or another proposed deployment, whether proprietary or open source. The spreadsheet includes four pages (or 'tabs') – the calculator, a worksheet for internal calculations (if required) and two examples.

Because every organization is unique and may have existing processes to use in financial analysis, it will be an organization's responsibility to determine whether this calculator can be used as is, needs to be modified, or should only be used as a model for calculating financial analysis using an existing tool. The calculator was created with the assumption that no internal financial processes are in place. In addition, not all of the cost elements listed in this report will apply to every project.

The associated spreadsheet can be obtained from The 451 Group's website or through a 451 account representative. The spreadsheet is available in Microsoft Excel format. The cells in the spreadsheet are general numbers to support international usage, whatever a given currency may be.

5.1 LAYOUT

The calculator includes time-based columns, with financial elements and results presented in rows. The columns include the initial investment period, three annual periods and the totals. The rows include the 10 most common hard costs, subtotaled in a hard-costs row, as well as individual rows for soft costs, internal costs and revenue. The remaining rows include information pertaining to the financial return – cash flows (both for the required period and cumulative), the rate of return by time period, when the payback occurs (payback period), the net present value, the internal rate of return and the cost of capital. There is also space for notes. If a review of these various metrics is required, they were covered in the 'Financial analysis 101' section of this report.

Figure 1 The Calculator

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**451 CAOS Report 2 - Cost Conscious**												
2	**A practical guide for understanding and calculating the financial benefits of open source for enterprise IT projects**												
3	**The 451 Group**												
4													
5	**Calculator - Example A**												
6													
7		**Open Source Option**						**Alternative/Existing Option**					
8		Initial Investment	Year 1	Year 2	Year 3	**TOTALS**			Initial Investment	Year 1	Year 2	Year 3	**TOTALS**
9	**HARD COSTS**	**76,250**	**41,200**	**111,150**	**49,100**	**277,700**		**HARD COSTS**	**296,250**	**99,200**	**176,150**	**104,100**	**675,700**
10	Software Licenses	-	-	-	-	-		Software Licenses	200,000	50,000	50,000	50,000	350,000
11	Hardware	25,000	3,000	3,000	3,000	34,000		Hardware	25,000	3,000	3,000	3,000	34,000
12	Support	-	25,000	30,000	35,000	90,000		Support	-	30,000	35,000	40,000	105,000
13	Development	22,000	2,500	5,000	2,500	32,000		Development	22,000	2,500	5,000	2,500	32,000
14	Professional Services	12,500	3,000	-	-	15,500		Professional Services	25,000	6,000	-	-	31,000
15	Training	7,500	-	-	-	7,500		Training	15,000	-	-	-	15,000
16	Testing	3,000	3,000	3,000	3,000	12,000		Testing	3,000	3,000	3,000	3,000	12,000
17	Operations	2,500	500	500	500	4,000		Operations	2,500	500	500	500	4,000
18	Staffing	-	-	65,000	-	65,000		Staffing	-	-	75,000	-	75,000
19	Maintenance Contracts	3,750	4,200	4,650	5,100	17,700		Maintenance Contracts	3,750	4,200	4,650	5,100	17,700
20													
21	**SOFT COSTS**					-		**SOFT COSTS**					-
22	**INTERNAL COSTS**					-		**INTERNAL COSTS**					-
23													
24	**REVENUE**		**250,000**	**500,000**	**800,000**	**1,550,000**		**REVENUE**		**250,000**	**500,000**	**800,000**	**1,550,000**
25													
26	**CASH FLOW**							**CASH FLOW**					
27	Period	(76,250)	208,800	388,850	750,900			Period	(296,250)	150,800	323,850	695,900	
28	Cumulative (Payback)	(76,250)	132,550	521,400	1,272,300			Cumulative (Payback)	(296,250)	(145,450)	178,400	874,300	
29													
30	**RATE OF RETURN**		607%	450%	1629%			**RATE OF RETURN**		252%	284%	768%	
31	**PAYBACK PERIOD**	**Year 1**						**PAYBACK PERIOD**	**Year 2**				
32	**NPV**	**954,643**						**NPV**	**591,892**				
33	**IRR**	**340%**						**IRR**	**82%**				
34	*cost of capital %*	12%						*cost of capital %*	12%				
35													
36													
37	NOTES:							NOTES:					
38													
39													
40													
41													
42	**November 2006 Edition**												

The associated spreadsheet can be obtained from The 451 Group website http://www.the451group.com/caos/451_caos_listing.php or through a 451 Group account representative. The spreadsheet is available in Microsoft Excel format.

5.2 TIME HORIZON

The time horizon for the calculator has been set at three years. This column can be increased to support additional years, however. Make sure to check the formulas in the spreadsheet when making any changes. While the formulas should automatically account for added time periods, accuracy depends on how the modification is done. We recommend not going past five years, due to the changing nature of technology and the useful life of an IT project. In contrast, doing a one-year financial analysis for a project may fail to show future financial benefits. It can often take more than one year for payback to occur. Three years is the recommended time horizon, although an organization or a specific IT project may have its own time horizon requirements. If there is a requirement to calculate financial benefits by quarters, or another duration of time, the spreadsheet will need to be modified. This will require modifying some of the measurements – especially NPV and IRR – that are based on annual periods.

The initial investment period, sometimes referred to as 'year zero,' is important for calculating both NPV and IRR. The initial investment period covers costs prior to a launch – the up-front costs. These would generally be startup costs, such as initial software licenses, hardware, maintenance contracts, development, professional services, training, testing, and possibly some staffing and operations. Year one begins after launch. With regard to a phased launch, the costs may need to roll out over multiple periods. The initial investment period could cover the first phase, with additional phases and ongoing development captured in the following time periods.

5.3 FINANCIAL ELEMENTS

The calculator is primarily focused on the quantification of hard costs over time. The financial elements that are included by default are the 10 most common hard costs, with additional rows for soft costs, internal costs and revenue. Custom financial elements may need to be added, and cost elements that are not relevant can be ignored or removed. If soft costs, internal costs or revenue are to be used in the calculator, then specific entries for these tracked financial elements will need to be added, either by modifying the calculator or by totaling the amounts on the worksheet page. If new rows are added, make sure that the 'totals' columns (marked in bold) include these new rows, and that the formulas in the spreadsheet remain valid. Three financial elements that are compounded into a single entry called 'internal costs' will be difficult to track without showing the calculations somewhere. Feel free to use the Worksheet page. Revenue is not used in the initial investment period, since estimated revenues will occur only after the project is deployed.

If depreciation, amortization or capital and expense budgets in a financial analysis must be accounted for, then ask someone in the organization how best to account for these. Depreciation may be best captured as an internal cost, or within the financial element that it relates to (hardware, for instance). Capital and expense budgets may be best dealt with as a note in the calculator or as a secondary calculation on the Worksheet page.

5.4 INPUT

Not every spreadsheet cell requires an input. Some of the cells are based on formulas and will be calculated automatically based on an input. The initial-investment column, along with the time-period columns, will require input down to the revenue row, but not beyond it, with the exception of the hard-costs row itself, which is the sum of all the individual hard costs. The information from cash flow and below is automatically calculated based on input from above. Amounts should be placed according to when they are incurred. This will show which costs are recurring or onetime, and which costs are fixed, variable or incremental in nature. A recurring annual support contract would have an amount listed in each time period of the calculator, while a onetime hardware cost would be listed when it's incurred (likely in the initial investment period). A financial element row may contain multiple onetime costs. Servers may be purchased during the initial investment period, and an additional server purchase may be made in a subsequent time period.

Every amount added to the calculator will be listed as a positive number. While this may be confusing when including both revenue and costs on the same spreadsheet, revenue is treated separately from the financial elements above, and the formulas in the spreadsheet take this into account.

If there are important notes to reference from within the calculator (e.g., assumptions, details, etc.), make sure to include them in the notes area. Don't forget there is a Worksheet page to allow for calculation prior to including items in the calculator.

5.5 COMPARISONS

The most common use for the calculator is to compare a proposed open source project with an existing implementation or a competing option. Thus, the calculator is laid out, by default, with room for a comparison of two options ("This is what the cost is today, and this is what the cost will be if we go with open source."). If more than two options will be compared, then copy one of the solution sets for as many additional options as needed.

When comparing an open source project to an alternative one, the information will align itself allowing for direct comparisons, assuming that the same time horizon is used and that the same costs for both proposed projects are captured. Cost avoidance and opportunity costs are best understood through comparing projects.

When comparing an open source project to an existing implementation, the comparison will need to be made with the costs per period, and probably not with the initial investment, taking the investment costs of the new project into account, of course. The initial investment for the existing implementation is a sunk cost, and it is not necessary to capture this in the spreadsheet. The money has already been paid to develop and deploy the existing implementation, and those costs are not recoverable. The proposed project, on the other hand, has new costs for development and deployment. What's most relevant in this type of comparison are the ongoing costs of the existing implementation compared to the ongoing costs of the proposed project, plus the initial investment of this same proposed project. If it costs more to develop, deploy and operate the replacement over the time horizon when compared to the operation of the existing implementation, then the replacement is not cost-effective.

If a comparison will not be used in the calculator, then changes need to be made to the calculator to include any cost reductions associated with the proposed project, otherwise this financial analysis will not effectively display anticipated cost savings. The best option in this case is to add another row above revenue called 'cost reductions' and capture any known cost savings associated with this project. Many of the formulas to take this information into account will need to be modified.

5.6 FINANCIAL RETURN

The financial benefits of a proposed project will be captured using the most common metrics. These include annual cost savings, annual rate of return, payback period, internal rate of return and net present value. An organization may prefer one metric to another. Each of these metrics provides distinct information, which is important to understand. The annual cost savings, while not listed within the calculator, can be derived by comparing multiple projects. This will indicate how much one option saves over another on an annual basis. The rate of return (or 'the annual rate of return' if time periods are annual) is how much, as a percentage, an investment returns annually. The payback period indicates how long it will take to pay back the initial investment for the project. The NPV and IRR indicate whether an organization would be better off investing in this IT project or another opportunity. When using multiple metrics, a more complete financial picture for a project will be gained.

The payback period row will indicate in which time period the payback for the initial investment occurs. If the payback period is shown as 'unknown,' then the payback does not occur during the time horizon and may not occur at all, even if the horizon is extended. A

negative IRR, indicating a poor financial investment, will show up as 'N/A' in the calculator. The NPV can be a positive or a negative return amount.

An organization's cost of capital must be figured, updating the default 12% used in the calculator for the calculation of the NPV. This number should be available from someone in the organization's finance department. If there is no set cost of capital rate for the organization, then continue to use the default 12%.

The cash flow of the proposed projects helps to determine the rate of return, the payback period, the NPV and the IRR – all forms of ROI. As for TCO, this can be derived by looking at the totals column, often in comparison with the TCO of another option.

5.7 THE FINDINGS

If use of the calculator shows a positive financial benefit from a proposed open source initiative, then congratulations… maybe. The alternative may provide an even better financial proposition. If this is the case, take this information into consideration along with the other benefits (nonfinancial in nature) that come from a proposed open source initiative. Remember that while achieving cost savings may be one of the top reasons for adopting open source, it's not the only reason.

But before coming to a definitive conclusion on the numbers, it is important to review the section on best practices for helpful ideas in making this financial analysis the most informed it can be. After reviewing the best practices, updates to the financial analysis (such as the use of a margin of error) may need to be made.

Section 6: Best practices

This section covers the best practices for approaching the issue of financial benefits of open source, the role of financial benefits in the overall justification effort, the calculation process, and follow-up issues such as presenting the findings and comparing actual costs and benefits incurred over time with the expected results.

There are numerous best practices relating to the financial analysis process that will help make a more accurate, compelling case for a proposed project. Please review these best practices before submitting the numbers for review. Some of these best practices will already exist within certain organizations.

Be consistent

Comparing multiple options can be difficult. These options are often dissimilar in nature. What keeps them comparable is consistency – consistency in approach, consistency in estimation and consistency in which financial elements are selected (or not used). When considering soft costs and benefits for an open source initiative, include them for the alternatives as well. The financial analysis is of no use to anyone if it's not balanced. The other aspect of consistency comes into play when comparing specific financial elements across multiple options. If the existing system has support services of one level and the costs of a completely new support level for a proposed project are being calculated, this may not be a fair comparison.

Double-check the math

While this may seem like an obvious best practice, when dealing with a lot of numbers and the process of building formulas (using a calculator created by someone else), it doesn't take much to introduce an error in the calculation process. Make sure to review all numbers at least twice before presenting the findings.

Show the math

Plan on using the calculator worksheet with all of the calculations for the relevant financial elements. Show all of the math, since these details may be needed or requested in the future. After a review by a CIO or CFO, there may need to be adjustments based on an organization's own practices. Generating a total amount for software licenses, for instance, does not help in understanding the different licenses that are required (or being eliminated). The other problem that will come up when not keeping detailed records is that when revising the calculations later, the manner in which the number was calculated may be difficult to determine.

The possible exception to this best practice is labor costs. When calculating labor costs based on salaries, which may be confidential, obscure the worksheet math to avoid people reverse-engineering individual salaries from the full calculation.

Keep the numbers up-to-date and share them

Building a financial analysis should not be considered a onetime event. Things change. Estimates may be incorrect; they should be updated with real-world numbers once these become clear. The scope of a project may change – and the costs and benefits will need to be updated accordingly. Revisit a financial analysis at least twice a year, if not more frequently. If there is a known cost change, ideally it should be updated immediately within a financial analysis.

Don't forget to share the updated financial analysis. Not being asked for an update is not a good reason to withhold this information. Many people forget to follow up on past projections of financial benefits. It is better to be open and honest, even if the financial benefits aren't as compelling in the end.

Gather the correct information

Don't assume that the six-month-old quote from a supplier is still valid. When doing the calculations, ask for updated numbers wherever possible.

Use a margin of error

When numbers are estimated, account for the fact that they may indeed be wrong. Mistakes will be made. Contingency was listed earlier as an internal cost. Regardless of whether your organization requires a contingency for a proposed IT project, factoring in a margin of error for individual costs is a good idea, even if you have included a contingency for the overall project.

The margin of error should be proportional to the level of uncertainty in the estimate. If there is only 50% confidence in an estimate, then double the estimated cost or halve the financial benefit, whichever it may be. This is especially important when dealing with soft costs and revenue projections.

Understand short-term vs. long-term issues

Being too focused on either the short-term or long-term costs and benefits may limit the ability to realistically gauge the success of a project. When too focused on the short term, a project may not yet have reaped the expected financial benefits. When too focused on the long term, it may be difficult to identify a failing project. The reality is that there are both short-term and long-term benefits, and that milestones along the way should be considered to track the relative success of a project.

This is why a multiyear calculation is recommended – with three years being ideal. Sometimes it will take a project a few years to bring the big payback. When too focused on the first-year results, it may be difficult to justify a project. When using a longer period – say, six years – it may exceed the useful life of the project, and the factors of inflation and depreciation become more important.

Time does have an effect on costs. An organization and its technology may change a great deal in six years. Too short or too long a time span used in the calculator are both problematic. When focused on the short-term gain, consider using a three-year projection to educate decision-makers on the longer-term benefits as well.

Recognize timing issues

Timing is key. Any number of costs for an existing implementation and the proposed alternatives are time-based – for example, support and maintenance contracts. The proper timing of contracts can save money, and the poor timing of contracts can present risks and, ultimately, more cost. If an organization is already planning to spend money for a technology refresh, such as new hardware purchases, it may be able to time a proposed project around this to avoid additional hardware costs.

The other important time standards are the organizational budget cycle and fiscal year. The proposed project may not be in a position to be considered for execution until some point in the future. The business case will need to be built, with the associated financial analysis, in preparation for the next budget cycle. It can be difficult to obtain new funds for a proposed project during the middle of an organization's fiscal year. Then again, there may already be IT budget allocation for the project, or there may be a more flexible funding situation in the organization.

When considering the timing of terminating old contracts and signing new ones, plan for some padding. Don't terminate old contracts in sync with a proposed launch date, since launch dates may change. Also, don't wait to sign new support contracts until the proposed launch date, since the support benefits during the development and testing phases of a project will be missed. This is often a period that sees great use of a support contract.

The other timing issue is a proposed launch date. Does an organization have a predictable business cycle? If so, plan a launch, if possible, corresponding to a low point in activity. For instance, in the retail sector, the activity curve is at its peak around the end-of-year holiday season, while in the travel sector, the activity peak is in the summer during vacations. Avoid the rollout during activity peaks, whenever possible.

This rollout timing may involve an end-of-life support issue, a forced migration or a requested vendor upgrade that is not cost-effective for an organization. A proposed open source project may be supported due to this pressure. If this is the case, make every effort to launch well before the product support of an existing system has ended. This will reduce risks.

Embrace standards

One of the values of open source is that open source components, as a general rule, are standards-based and can be swapped out. As Martin Fink says in his book 'The Business and Economics of Open Source,' standards are a key aspect of keeping costs low. Embracing standards will reduce any future switching costs and dependencies on vendors. Moving a standards-based code from one component to another requires far less work than changing technologies, and a team trained on a common set of standards will have little or no training needs. This will also likely reduce future recruiting costs and will result in a larger pool of candidates to choose from.

Standardization leads to commodification. When using a commodity platform (especially in terms of hardware), costs and potential for reuse go way up. Don't underestimate the value of standards-based technology. It will be a source of financial benefit in the future, even if this is difficult to quantify in the short term.

Review the contracts

Anytime a technology change is proposed, it is a good opportunity to go back and review the legal contracts, ideally with the aid of an organization's legal counsel. Are there any termination penalties for changing technology or suppliers? Is there a required notification period? Multiyear contracts will present the greatest challenges. Any penalties or future payments that are directly related to a proposed project need to be captured as costs.

Identify suppliers

With open source, it's not often clear who the best providers are for the services required, such as support, professional services or training. With open source, there will be more choices than with proprietary software. Don't assume that the first supplier contacted is the best supplier available.

The number of choices may require a request for information or a request for proposal. While this topic is not covered in this report, there are plenty of good sources for this information online, and an organization may already have an established process for RFIs and RFPs. The other benefit to an RFI/RFP is that it clarifies the requirements and goals.

When exploring self-support as an option, the internal team should respond to the RFI/RFP as though it were a provider. This is the best way to compare self-support with other options.

Build a business case

As mentioned before, the financial analysis for an open source initiative should not be considered the entirety of a business case for open source adoption. The calculator should not be the only exposure that IT leadership has to the open source proposal. The results of the calculator should be coupled with a wider business case for open source. The primary benefits of using open source in an organization may not be financial in nature. We have listed a number of resources for building a business case for open source in the 'Resources' section of this report.

Approach benefits beyond finances

Although the focus of this report is on the financial benefits of open source, don't lose sight of the fact that there are other tangible benefits to an organization that may come from adopting open source. While the financial benefits may actually be meager or nonexistent in some cases, put this information in context with the other benefits for the proposed project.

Know the organizational requirements

Understand the project and funding approval process within an organization and what format the financial analysis information should be presented in. Are there any corporate standards or forms that must be used? The material generated from the calculator may need to be placed within a different format.

Know the decision-makers

Know who in the organization will be reviewing the financial analysis. If possible, speak with all of these people regarding their expectations. The CIO and the CFO may be expecting different information. The CFO will likely be more process-focused than the CIO, who is more sympathetic to the strategic value and intangible benefits of technology initiatives to the organization.

Consider a pilot project

There is value in an organization proceeding with a pilot project when there is little or no in-house experience with open source. The use of a pilot project coupled with financial analysis reduces risks, helps in understanding the true costs of an open source deployment and determines what other benefits may come from open source in an organization. The incremental adoption of open source, while more time-intensive, is the most prudent course of action.

As the experience in implementing open source within an organization grows, the overall time to implement and cost to implement will both decrease, and the ability to estimate costs will improve. A first open source initiative may be the most expensive one in terms of overhead as an organization gains familiarity with the intricacies of open source.

As for determining the best option for a pilot project, try to avoid selecting a critical production project for a pilot. Maria Winslow, in her book 'The Practical Manager's Guide to Open Source,' says that "any noncritical system that is transparent to end users is a good choice." Start small, and monitor the project (and costs) closely. The trade-off with a pilot project is that the financial benefits may be less dramatic.

Understand the nuances of various project types

Not all projects are created equal. When comparing an open source deployment to an existing one, the financial elements are not always similar. The same can be said about comparing any two prospective projects – even two open source initiatives. Projects don't have to be treated identically, as long as they are treated consistently. Each project will have unique financial elements at play.

When looking at the replacement of an existing system, understand that migrations can be disruptive, and are often costly. The status quo is often at a financial advantage. If this is a forced or required migration, a comparison of the open source option to another option is ideal. The other important point to consider here is that if the migration is not mandatory, any money or time spent planning the migration really should be treated as costs.

When looking at a new project not involving the replacement of an existing system and comparing an open source option with a proprietary one, there will likely be significant financial benefits with the open source option due to the savings from software licensing fees, assuming that all other elements are equal.

Section 7: Examples

This section presents example uses of the calculator. These focused 'case studies' will show various outcomes, based on the unique aspects of each example. The primary purpose of this section is to provide concrete use examples for the calculator and to express the diversity of outcomes that are possible when using the calculator (not everyone saves money with open source). Please review the examples within the spreadsheet as this section is reviewed.

Example A: E-commerce initiative

Company A has a new consumer e-commerce initiative and is attempting to determine whether to use an open source application server or a proprietary alternative. Company A uses a three-year time horizon, with the default cost of capital at 12%. Company A does not allow for the quantification of soft costs. The revenue projection for the e-commerce initiative is included at equal rates for both options, since a revenue variation is not expected based on the technology used.

When compared with the proprietary option, the open source alternative does not include software license costs, thus the row is left blank. The hardware costs are identical in both options, and are listed (although this row could be removed, if preferred). In this example, there is a reduction in the overall support, training and professional services costs. While a reduction in support and training costs isn't always the case with open source, such a reduction can often be seen with regard to infrastructure software.

From the results, it is clear that the open source option provides a greater value than the proprietary one, although both options result in a positive return, based on the expected revenue. While the company would ultimately generate a positive return using either option, it would benefit from a net present value of greater than 350,000 more by selecting the open source option, and the payback would occur during the first year (as opposed to the second year). The cost difference in the two options is due in large part to the software licensing costs of the proprietary offering. This is clear from the costs associated with each initial investment period – a difference of over 200,000.

Example B: Desktop migration

Company B has had success with a Linux desktop pilot project in its IT department. It would like to compare the costs of a companywide Linux desktop migration to the existing proprietary system. Company B uses a five-year time horizon, although the goal of the project is to generate cost savings over the existing system in less than three years. The cost of capital has been set at 15%. Company B does not allow for the quantification of soft costs, although some internal costs have been included (in the form of a 10% project budget contingency). There is no revenue anticipated directly as a result of this project. The initial investment for the existing system is not included, since this is a sunk cost and is not recoverable.

Company B has decided to use its own internal distribution of Linux, thus eliminating software licensing costs in favor of greater development, professional services, testing and operations

costs. By bringing the support structure in-house, the company will need to hire two experts to augment the support team. Both options in this example include planned software upgrades in year two and year four, although the proprietary system includes software upgrade costs.

When comparing the existing system with a proposed open source deployment, the rate of return and the payback period cannot be determined automatically, since there is no revenue to consider. In this example, the two options must be compared manually. When looking at the five-year time horizon for the Linux desktop migration, the existing proprietary system is more compelling from a cost perspective. At the end of the five years, the Linux desktop migration would cost the company a net present value of around 300,000 more, and therefore it fails to provide cost savings in less than three years, as requested. In the long run, the Linux desktop migration would be the most cost-effective option, but this benefit occurs beyond the five-year time horizon. The short-term disadvantage is that there is an initial investment of approximately 200,000 that does not occur with the existing system (since this involves sunk costs). For the Linux desktop migration project to be approved, management must take a longer-term view of the financial return and be willing to pay for the initial investment. The alternative would be to postpone the migration until the current proprietary system was no longer supported and an upgrade was mandatory.

Section 8: A survey of end users

In association with this report, we conducted a survey on the topic of open source cost savings. The purpose of this survey was to collect information on the role of cost savings in open source software adoption. The survey was not intended to be statistically significant. Rather, it was designed as a final check of our analysis of user perceptions based on regular conversations we have with the end-user community as part of the 451 CAOS service.

In order to participate in this survey, we asked that each respondent have an understanding of their organization's open source support efforts and an awareness of the IT project approval process, specifically the financial analysis. While the main focus of this survey was enterprise IT end users, software vendors that support open source internally were invited to participate as long as their responses related specifically to internal open source support – where they were acting as users – and not to products or services provided to their customers.

The survey ran for a period of 47 days, from September 11 through October 27, 2006. During this time, we received 105 unique responses. The survey was promoted through our 451 CAOS Theory blog and through targeted mailings to open source end users and software vendors. The survey included nine questions and an optional request for follow-up and contact information.

8.1 SURVEY RESULTS

We took the results of this survey and combined them with information gathered in conversations with end users initiated specifically for this report, plus our experience in talking to vendors and their customers over the past few years. Here are some of the main points raised by the survey:

- Lower cost remains one of the top reasons for open source software adoption. In this survey, it was the top reason, followed by reduced dependence on vendors and then flexibility. The results of our survey match other survey results on the same topic. We do not believe that cost savings as a major motivator for open source software adoption is going away anytime soon. Three out of four respondents indicated that cost savings were an important factor in the decision-making process for open source software adoption.
- Historical surveys on this topic have broken down responses by organizational role and found that executives tend to list lower cost as their top reason for open source software adoption, while software developers tend to list flexibility as their top reason. Much to our surprise, our survey did not find a similar pattern, with 'lower cost' being equally distributed across organizational roles. We believe this is due in large part to greater awareness and value of the organizational motivators for open source adoption, even if cost savings are not the most important consideration to a software developer.
- After the adoption of open source software, flexibility climbed to match cost savings as the greatest benefit. While achieving cost savings may be the strongest motivator for adoption, it is not always the primary benefit upon project completion. After flexibility and lower cost, users listed reduced dependence on vendors as the next biggest benefit.

So the top three reasons for open source adoption remained the same after adoption, but they shifted somewhat in priority.

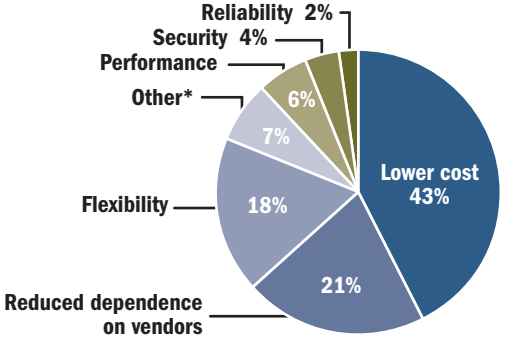
- Almost three out of four respondents listed software licenses as the greatest opportunity for cost savings, followed by maintenance contracts and license management. As the cost savings for software licenses tend to be more dramatic for new projects or large-scale growth of existing projects, this raises some potential issues relating to smaller-scale projects or revisions to existing open source projects, once the major source of cost savings is gone.
- More than a third of the respondents indicated that their organizations do not have formal processes in place for IT project financial analysis. Of those respondents that did, the majority listed multiple methods of financial analysis used. This reinforces our multi-method approach to the calculator used in this report, which aims to incorporate the various financial analysis practices used by organizations.

Figure 2 – Survey results

Consideration of cost in the open source decision process

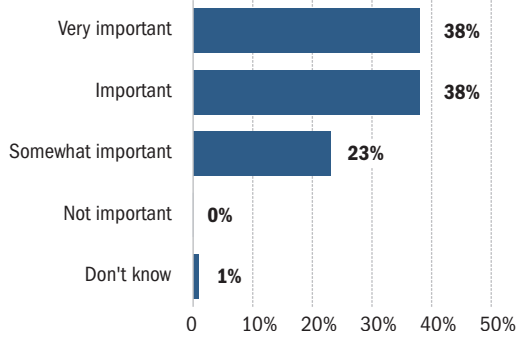
Cost savings is no. 1 reason for using open source.

What was the primary reason that your organization decided to use open source software?



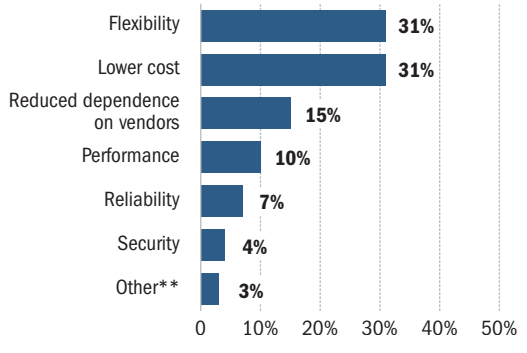
"Cost savings" is important before adoption...

Overall, how important was the issue of cost savings in the decision-making process for open source adoption?



...but not as important after adoption.

After your organization adopted open source software, what was the primary benefit of its use?

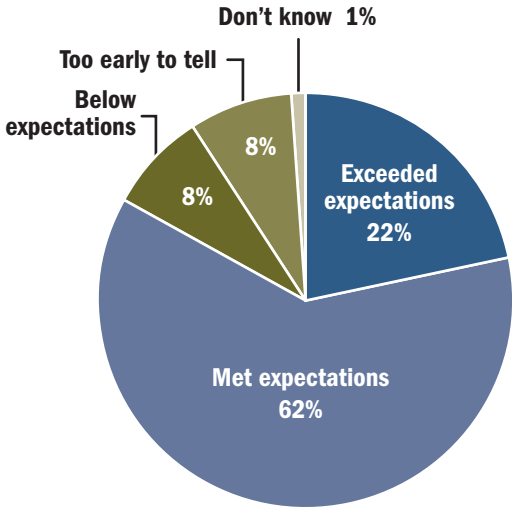


* 'Other' responses included: quality, improved market competition, ability to integrate with other standards-based software, more stable long-term upgrade path and source code, functionality, and being the only product for a specific requirement.
** 'Other' responses included: frequency and availability of updates, functionality and social reasons.

Evaluation of open source projects

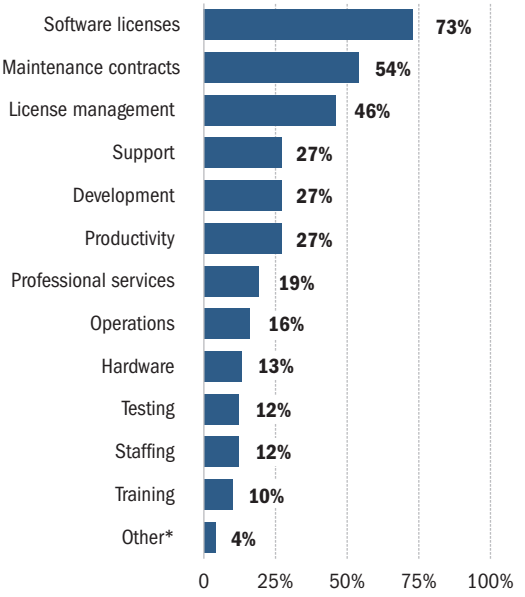
Cost savings met or exceeded expectations...

To what extent were cost savings achieved through the use of open source software?



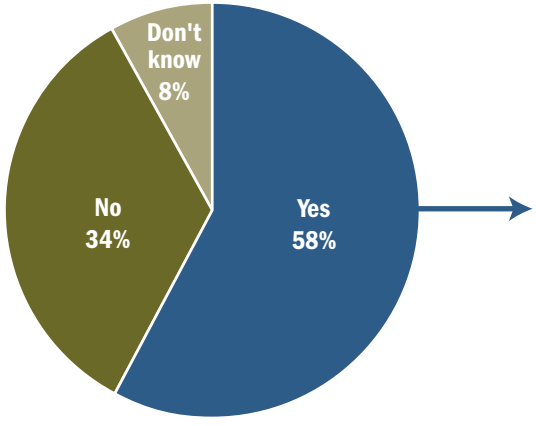
...by reducing software and maintenance expenses.

Where do you believe the cost savings will come from (or, if this information is already known, where did the savings come from)? (Multiple selections allowed per respondent.)



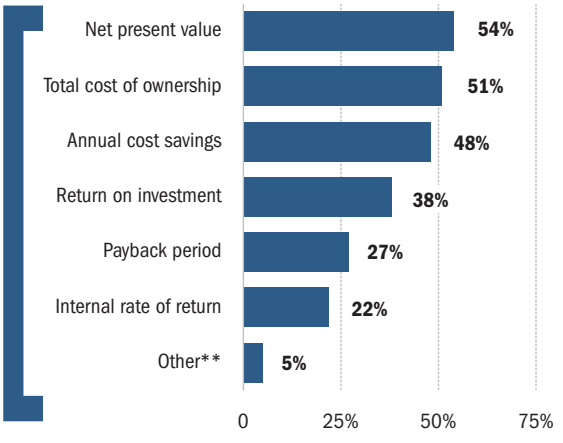
Most evaluate IT projects formally...

Does your organization have formal processes in place for IT project financial analysis?



...using a variety of financial analysis methods.

If you answered 'yes' to prior question, which metrics do you use for financial analysis? (Multiple selections allowed per respondent.)



* 'Other' responses included: upgradeability and accessibility, time to deployment, security, and accumulation of customizations and upgrades. ** 'Other' responses included: Earned value metrics, 'social ROI' and SOW-based consulting engagement.

Survey respondent profile

Role in organization	Percentage	Total
C-level executive	26%	27
Director	20%	21
Software architect	18%	19
Manager	16%	17
Individual contributor	13%	14
Vice president	7%	7

Type of organization	Percentage	Total
Enterprise end user	59%	62
Software vendor	22%	23
Systems integrator	12%	13
Government	7%	7

Section 9: Resources

We have listed various resources that were used in researching this topic. These resources can be reviewed for additional details.

9.1 BOOKS

Berkun, Scott. (2005). **The Art of Project Management**. Sebastopol, CA: O'Reilly Media.

Brooks, Frederick P., Jr. (1997). **The Mythical Man-Month**. Reading, MA: Addison Wesley.

Fink, Martin. (2003). **The Business and Economics of Linux and Open Source**. Upper Saddle River, NJ: Prentice Hall.

Kavanagh, Paul. (2004). **Open Source Software: Implementation and Management**. Burlington, MA: El Sevier.

Winslow, Maria. (2004). **The Practical Manager's Guide to Open Source**. Chapel Hill, NC: Open Source Migrations.

Woods, Dan, & Guliani, Guatam. (2005). **Open Source for the Enterprise**. Sebastopol, CA: O'Reilly Media.

9.2 ARTICLES

D'Agostino, Debra. (November 23, 2005). *CIO Insight Research Study: Open Source Turns Strategic*. **CIO Insight**.
http://www.cioinsight.com/article2/0,1540,1893081,00.asp

McAllister, Neil. (January 9, 2005). *You can't kill TCO*. **Computerworld**.
http://www.computerworld.com.au/index.php/id;1067744063;fp;16;fpid;0

(May 23, 2005). *Managing Projects*. **Baseline**.
http://www.baselinemag.com/article2/0,1397,1817374,00.asp

Melymuka, Kathleen. (August 1, 2005). *Quick Hits*. **Computerworld**.
http://www.computerworld.com/softwaretopics/os/linux/story/0,10801,103553,00.html

Raymond, Eric S. (August 2, 2002). *The Magic Cauldron*.
http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/

Scavo, Frank. (May 22, 2005). *Key Advantage of Open Source is Not Financial Benefits*. **Computer Economics**.
http://www.computereconomics.com/article.cfm?id=1043

Zachary, Raven. (March 2006). *Six Options For Open-Source Support*. **Optimize**.
http://www.optimizemag.com/showArticle.jhtml?articleID=180207025

9.3 BUILDING A BUSINESS CASE FOR OPEN SOURCE

The following are resources on the topic of building a business case for open source (not fully covered in this report).

Fink, Martin. (2003). **The Business and Economics of Linux and Open Source**. Upper Saddle River, NJ: Prentice Hall.

Goldman, Ron & Gabriel, Richard P. (2005). **Innovation Happens Elsewhere: Open Source as Business Strategy**. San Francisco, CA: Morgan Kaufmann.

(n.d.). *Open Source Case for Business*. Open Source Initiative.
http://www.opensource.org/advocacy/case_for_business.php

(n.d.) *Pattern Book for Open Source in the Enterprise*. **Flashline**.
http://wiki.flashline.com/wiki

Winslow, Maria. (2004). **The Practical Manager's Guide to Open Source**. Chapel Hill, NC: Open Source Migrations.

Woods, Dan, & Guliani, Guatam. (2005). **Open Source for the Enterprise**. Sebastopol, CA: O'Reilly Media.

Section 10: Methodology

To complete this detailed and comprehensive research report on open source cost savings, The 451 Group used a variety of information sources. The most important was a series of interviews with users and vendors. This research was supplemented by additional primary research, including attendance at a number of trade shows and industry events, as a well as a special survey on the topic. This research was conducted in both the US and Europe.

Reports such as this one represent a holistic perspective on key emerging markets in the enterprise IT space. These markets evolve quickly, though, so The 451 Group offers additional services that provide critical marketplace updates. These updated reports and perspectives are presented on a daily basis via the company's core intelligence service – the 451 Market Insight Service. Perspectives on strategic acquisitions and the liquidity environment for technology companies are updated on a weekly basis via the company's forward-looking M&A analysis service – 451 TechDealmaker – which is backed by the industry-leading 451 M&A KnowledgeBase.

The emerging-markets picture is further complemented from the end-user/customer perspective via a growing group of adoption research services – including the 451 Grid Adoption Research Service (GARS) and the 451 Commercial Adoption of Open Source (CAOS) Research Service. All of these services, which are accessible via the Web, provide critical and timely analysis specifically focused on the business of enterprise IT innovation.

This report was written by Raven Zachary, Senior Analyst and Open Source Practice Head, together with Lee Bruno, Editorial Director, Special Reports.

Any questions about the methodology should be addressed to Raven Zachary at:

raven.zachary@the451group.com.

For more information about The 451 Group, please go to the company's website:

www.the451group.com.

About The 451 Group ■ The 451 group is an independent technology-industry analyst company focused on the business of enterprise IT innovation. The company's analysts provide critical and timely insight into the market and competitive dynamics of innovation in emerging technology segments. Clients of the company – at vendor, investor, service-provider and end-user organizations – rely on 451 insight to support both strategic and tactical decision-making for competitive advantage. ■ The 451 Group is headquartered in New York, with offices in key locations, including San Francisco, London and Boston. The company also operates Tier1 Research – an independent division of The 451 Group, headquartered in Minneapolis – which analyzes the financial and industry implications of developments impacting public and private companies within the IT, communications and Internet sectors. ■ For additional information on the company or to apply for trial access to its services, go to: www.the451group.com.

Analyzing the business of enterprise IT innovation

New York
137 5th Avenue (between 20th and 21st Streets)
12th floor
New York NY 10010
Phone: 212 505 3030
Fax: 212 505 2630

London
37-41 Gower Street
London, UK WC1E 6HH
Phone: +44 (0)20 7299 7765
Fax: +44 (0)20 7299 7799

San Francisco
One Kearny Street
Suite 400
San Francisco CA 94108
Phone: 415 989 1555
Fax: 415 989 1558

Boston
52 Broad Street
2nd Floor
Boston, MA 02109
Phone: 617 261 0699
Fax: 617 261 0688