lustratus

white paper

# The Open Source Value Proposition for SOA

Lustratus Research Limited
St David's. 5 Elsfield Way, Oxford OX2 8EW, UK
Tel: +44 (0)1865 559040

www.lustratus.com


# Table of Contents

# About Lustratus Research

Lustratus Research Limited, founded in 2006, aims to deliver independent and unbiased analysis of global software technology trends for senior IT and business unit management, shedding light on the latest developments and best practices and interpreting them into business value and impact. Lustratus analysts include some of the top thought leaders in market segments such as service-oriented architecture (SOA) and business integration.

Lustratus offers a unique structure of materials, consisting of three categories—Insights, Reports and Research. The Insight offers concise analysis and opinion, while the Report offers more comprehensive breadth and depth. Research documents provide the results of practical investigations and experiences. Lustratus prides itself on bringing the technical and business aspects of technology and best practices together, in order to clearly address the business impacts. Each Lustratus document is graded based on its technical or business orientation, as a guide to readers.

# Terms and Conditions

# Executive Summary

This paper is aimed at IT managers and architects considering the use of Open Source Software within a SOA project. The deployment of any software into an organization has implications far beyond a simple technical evaluation. Once deployed it can be expensive, risky and disruptive to remove it. But relying on obsolete or hard to maintain software is equally worrisome. Therefore, the decision to deploy any new software requires careful analysis of the benefits, risks and costs associated with potentially relying on that software for many years to come. While Open Source Software (OSS) is a high profile topic in the software industry, a major inhibitor to adoption remains a lack of knowledge of how to evaluate the benefits and risks associated with its use. This lack of information leads organizations to either ignore OSS when it is the best solution or adopt OSS without fully understanding the consequences.

This paper focuses on the use of OSS to implement Service Oriented Architecture (SOA); an area which is notable by the number and popularity of Open Source projects. The goals of the paper are to clear the confusion around SOA Open Source Software and to provide decision-makers with the knowledge required to analyse whether an OSS-based solution is the best option for their SOA project. Furthermore, it highlights the value provided by OSS vendors which in many cases are crucial to the long-term success of OSS adoption.

As such, this paper should be considered as a guide for management in any organization that is in the process of selecting infrastructure software for use as part of their SOA program. It analyses the benefits derived from using Open Source Software for SOA projects: focusing in particular on the risk and budgetary dimensions. It clarifies the role of the businesses which provide services around the Open Source projects (the OSS vendors) and distinguishes between vendors providing OSS versions of closed source products and those providing OSS only solutions. Note that any reader unfamiliar with the concepts of Service Oriented Architecture and how it fits into the evolution of enterprise systems should read Appendix A prior to the rest of this paper.

This paper adopts the following structure:

1. Highlights the key issues that need to be addressed when building a SOA business case: The difficulties associated with funding SOA projects and the risks associated with such projects.

2. Explores the current state of Open Source Software acceptance in the enterprise which is both more limited and more extensive than generally perceived.

3. Explains how OSS can address the SOA business case issues by providing a model for software adoption well aligned with the roll-out of integration solutions.

4. Highlights the risks associated with OSS adoption and explains how vendors backing these projects mitigate these risks.

5. Explores the different elements of the value proposition of the OSS vendors and how to evaluate them.

## What is SOA?

SOA stands for **S**ervice-**O**riented **A**rchitecture, where IT programs and resources are encapsulated and made available as business services, each enacting a discrete business function such as 'Get Customer Details'. The services can then be used and reused as building blocks, assembled together into other business functions and processes as required.

## Open Source and Closed Source Software

**Open Source Software** (OSS) is software whose source code is available at zero cost under licenses such as the *Gnu General Public License* (GPL) which give the right to access, modify and distribution rights. Open Source software is typically available for download from the Internet.

**Closed Source Software** is the term sometimes used to contrast OSS with software available under traditional commercial licenses which do not allow access to source, modification or onward distribution rights.

## *Summary of findings*

OSS addresses the following problem areas for SOA:

| SOA Problem Area | OSS Derived Solution |
|---|---|
| SOA projects can be hard to fund as they are typified by large upfront license requirements, hard to execute funding models such as internal charge-back and uncertainty about the eventual functionality requirements and scale | The OSS business model promises low upfront cost, the ability to scale out and scale up usage. This allows investment to match requirements and removes some long term uncertainty around the project cost. Scaling out is possible because deploying onto additional servers has zero associated license cost. Scaling up is possible because OSS SOA projects follow a modular approach which allows the selection of required modules as and when they are needed. |
| SOA deployments will be long-lived and business critical. This increases the risk associated with vendor or product failure. | The Open Source model of freely available source and user community involvement mitigates the risks associated with vendor failure and product abandonment. However, this mitigation relies on the existence of an active user community or internal technical resources capable of taking on support and development of the project. |
| Skills required to use most integration products are hard to find and can be expensive to retain over the life-time of a SOA deployment. | OSS projects leverage standards heavily which makes it easier to find knowledgeable developers. As the software is freely downloadable, staff can try it out and build skills before committing to the project. The community of developers that surrounds the project also helps to foster expertise and provides a source for expert assistance. However, it should be stressed that many skills can only be built from real world experience of building large scale deployments and such skills are independent of whether open or closed source software is used. |
| All integration projects including SOA projects require large amounts of customization and integration into diverse 3rd party technologies. | OSS SOA projects are typically modular in structure with customization as a key design goal. Their open architecture and use of community development makes integration into 3rd party software easy and increases the potential availability of adapters. |

However, Open Source also introduces risks and costs not present with Closed Source software which seriously undermines the overall benefits of adopting OSS. Therefore, while Open Source Software may provide attractive business benefits in the context of SOA, serious consideration should be given to these issues. One approach to mitigating these is to partner with an OSS vendor. The benefits that may be derived from such a relationship include:

| Drawback of OSS | Value delivered by OSS vendor |
|---|---|
| All OSS SOA solutions rely on multiple OSS projects as no one project provides the complete functionality. The Open Source development methodology leads to multiple versions and potential incompatibilities across these required projects. Closed software vendors provide integration within their product set out of the box. This integration activity typically represents more than 20% of the vendor's overall development cost. | Most SOA OSS vendors provide a certified set of projects which are tested to work together. This provides the user with a robust platform to work with and a *'throat to choke'* if bugs are discovered. It is of course essential that the OSS vendor keeps the platform up to date with the latest OSS project features. |
| Without vendor engagement, OSS projects rely on users to drive innovation and sustain the development of the project. | OSS vendors have a current and on-going commercial stake in keeping the project competitive and vital. This can be through support for the OSS community or through direct investment in the development of the software itself. |
| Because OSS projects rely on standards, the skills required to ***use*** the projects are more available. However, the skills required to ***develop*** the projects themselves are often deeply technical and may be hard to acquire or hire. | OSS vendors committed to specific projects will develop the deep skills in-house and make these available to their customers. |

Finally, any analysis of the value of a partnership with an OSS vendor must recognize and take into account that OSS vendors differ in two key respects:

### *Business Strategy:*

In the SOA OSS space, the OSS projects promoted by the OSS vendors are very diverse. While all OSS vendors combine a number of projects to create their solution, not all are attempting to create a complete OSS-based solution: It is common with the larger vendors in particular to promote OSS as a light weight alternative to their *full strength* closed source products. For these vendors, it is essential that due diligence verifies that the OSS solution will be sufficient for all current and future requirements. If this is not the case, the cost of the closed source product must be factored into the business case.

### *"Product" Strategy:*

The SOA OSS vendors follow significantly different *"product"* strategies. Not only do the vendors promote and support different projects, they also place different emphasis on elements such as technology innovation, community-driven development, the integration of the projects into a coherent platform and enterprise support. Therefore for SOA, the different OSS solutions available need to be evaluated like any closed source product with consideration of both current features and roadmap. This is a very different situation to commoditized Open Source technology areas such as Operating Systems.

### *Open Source Software vendors*

While Open Source Software is freely available at no cost, there are still requirements for additional services such as support, maintenance and consulting around the projects. These services are provided by businesses which are typically referred to as Open Source Software Vendors. Open Source Software Vendors may also sell Closed Source Software (such as IBM), be exclusively providing Open Source Software (such as Red Hat) or provide general consulting services as well.

# *Building the business case for SOA*

The details of any IT business case vary from organization to organization. With SOA, this is particularly the case because Service Oriented Architecture as an enabling architecture will almost always be a secondary element within the business case: The primary element will be the solving of a business problem. The solution may benefit from SOA but the solution is unlikely to be directly attributable to SOA. Second of all, as an architectural model, SOA can be applied to solve a broad set of diverse problems. The problems which when solved will deliver the most value will vary with the scale of the organization and the industry it operates within. SOA has proven well suited for portal integration and integration of back-end systems. It is applied as the basis for data exchanges, and client-server systems. In each case, the SOA principles are common. What differs is the manner in which these principles are applied. Finally, the way the SOA principles are manifested in any given company reflect that organization's goals, IT strategy structure, history and even its corporate culture. For instance, some organizations are fundamentally federated while others are centralized: The SOA deployment should reflect these structures and should not introduce a tension between IT architecture and business organization. After all, SOA is all about alignment of technology with the business.

While the specifics of any business case may vary greatly, some elements will always be present. These are the key questions that must be answered as any case is built and the return on investment calculated. Questions such as:

**The roadmap:** How will we achieve the end-goal and what are the steps along the way? How long will the output from the project be used? How will the transition from development to deployment be handled and how will the project be sustained after development is completed?

**The benefits of completion:** What business benefits will accrue if the project is completed? What is the impact of not doing the project?

**The risk assessment:** What are the risks associated with the project and what are the risks associated with not proceeding? What are the longer term risks associated with the completed project? Will the project be able to meet the organization's need throughout its project life-span?

**The total project cost:** How much will the project cost including license fees, development costs, additional hardware, maintenance fees and on-going maintenance cost?

These questions will be discussed in greater detail later in the paper in the context of how OSS supports the SOA business case and how OSS vendors are needed to address OSS specific limitations. For now it is sufficient to point out that the issues that need to be addressed for a SOA project are in fact similar to those associated with any integration project with the aspiration to be rolled out across the organization. These risks are summarized below:

| Area of SOA risk | Explanation |
|---|---|
| Need for organizational maturity | SOA in particular requires a high degree of maturity around governance to ensure that the necessary policies around service and data definitions and use are complied to. For organizations unused to such governance, it can be particularly challenging to set-up governance mechanisms that will simultaneously control behavior and engage and involve stakeholders across the organization. |
| Lifespan and criticality of the deployment: | Successful integration deployments are often automated key business processes and have lifetimes from a minimum of 3 years. Typical life spans are 5-9 years with many systems continuing indefinitely. Will the software still be around throughout the life span of the deployment? |

| Area of SOA risk | Explanation |
|---|---|
| Degree of customization required to support the business requirements | Integration products have a deserved reputation for requiring a high degree of customization in each project. To a degree this reflects the underlying uniqueness inherent in each project. All customization must be maintained throughout the life-time of the deployment and necessitates retention of specialist skills and knowledge. |
| Degree of change expected during the deployment's lifetime: | The operating system and hardware required has an obvious impact but it is an aspect already understood by IT operations managers. The other dimension is changes in the function of the deployment. Clearly, all businesses change over the average lifetime of an integration deployment. Integration software deployments are in particular impacted by this change and it is unlikely that a deployment will remain useful and unchanged through the lifespan of the project. Therefore at the function level, the deployment must either be designed to change or it must be possible to extend the deployment by adding on additional components. |
| Availability and expertise of skills: | Any software deployment requires skills from both a development and a maintenance perspective. Any skills that are specific to individual products or OS projects carry additional risk as such skills may become hard to source and hence more expensive. |

## *SOA and budget Catch 22*

The fact that SOA only delivers its full potential when it is deployed across the entire enterprise leads to the greatest challenge when building the business case: Budgeting for the upfront and on-going technology investment. The classic approaches to funding IT infrastructure projects from central sources or on a project by project basis both face significant challenges. These challenges deter some organizations from adopting SOA at all or lead to high-risk or sub-optimal adoption strategies.

From its conception, it is clear that with SOA, the greatest benefits accrue in terms of reuse, agility and cross-organizational efficiency when it is rolled out broadly. However, such a widespread deployment is not an easy journey for any organization to make and will normally take a number of years across many individual projects for even the most committed organization. Furthermore each individual project will have its own focus. This leads to two major problems with funding a SOA initiative:

**Measuring tangible benefits**: SOA as with most infrastructure investments delivers benefit indirectly through enhancing and enabling the desired business driven outcomes. Including such indirect benefits with a SOA business plan is difficult and requires either co-opting of specific business projects as SOA projects or acceptance of less precisely measured benefits.

**Predicting the requirements over the life-time of the SOA deployment:** Integration software is expensive to roll-out across any organization and by its nature, the return on investment improves the more widely the software is deployed. Early on in the SOA roll-out, decisions must be made as to how the SOA infrastructure will be implemented: whether to use an ESB or extend the use of deployed EAI products, whether to use a SOA-specific management platform or to use an existing management platform, whether to buy a SOA registry or build a custom registry and so on. This does not mean that specific products must be selected and committed to for use across the organization: There will be requirements to use other infrastructure software in some situations where the technology base is different (e.g. for mainframe-based systems). In other situations, there will certainly be requirements for additional functionality to solve more complex requirements or less functionality to solve simpler requirements. However, the front loading of the decision and hence cost has always been problematic for any infrastructure project.

**Estimating the total cost of SOA:** At the outset it is hard to estimate how widely SOA will be deployed, at what rate the deployment will occur and what precise functionality

will be required across the set of business problems that may need to be solved. When using Closed Source software, each of these dimensions has implications on costs in terms of software licenses as well as effort expended on development and maintenance.

These issues create a *"Catch 22"* scenario when deciding whether to fund centrally or on a project by project basis:

**Centrally funded SOA:** Centrally funded SOA has the advantage of making it easier to ensure appropriate architecture choices are made, consistency is maintained and governance structures and processes are put in place. However other challenges emerge associated with the two funding models that can be used to cover the costs: Charge-back and central funding. Charge-back involves recouping the centrally borne costs from departments as they use the software. Unfortunately, this encourages departments to ignore the central SOA architecture and to go their own way. The chargeback cost on its own is unlikely to be used as the reason for this independence. However, it may be used to justify alternatives as having lower cost and hence higher return for that specific project. If SOA is funded centrally without chargeback, the large and hence very visible investment will struggle to be justified as the benefits will be speculative and hidden within business line projects which succeed because of the SOA program. If each department is asked to contribute outside of any specific business-drive project, it is hard to gain consensus across the departments as there is a perception that the rewards of all cooperating will be the grabbed by central IT although the risk is spread. This leads to a strong tendency for departments to take different paths which suit their own goals and the consistency breaks down.

**Project-level funded SOA :** The project funded approach tends to result in a project specific SOA: Each project creates a version of SOA well suited to its own needs and not suited to the organization's over-arching needs. Of equal significance to the long term success of SOA, a project level funding model will not be able to put in place the organizational level governance structures and processes required for longer term SOA success.

This problem with the business case has resulted in the following common scenarios:

### *High visibility/high risk SOA:*

In a minority of organizations, the argument for SOA can be made and backed as a strategic requirement at the highest level of the organization. The business case is often argued as part of a strategic need to decrease time to market rather than tied to benefits associated with specific goals. These organizations will tend to have a high degree of maturity in IT governance and operate in centralized businesses with considerable dependence on and understanding of IT for day to day operations (such as financial services firms involved in trading or large scale logistics firms with sophisticated tracking systems). Organizations are unlikely to take this approach unless the CIO is a true visionary CIO.

### *Grass roots developer-led SOA:*

Often closely aligned to the adoption of Web Services, SOA principles are introduced to existing projects rather than explicitly included within the business case. This is also known as *viral* SOA or *guerrilla* SOA. Because it makes SOA invisible, the tactical approach to SOA clearly diminishes the opportunity to extend the benefits across the organization and risks the creation of islands of SOA. However, the upside is that SOA is tried on already funded projects to solve real business problems. Once the project has been successful, the SOA aspect can be revealed and used to build more general use of SOA. While difficult, if the transition from grass-roots to enterprise wide SOA can be made effectively, such SOA programs can be highly effective.

### *Wait and see SOA:*

SOA may be ignored or tentative pilots undertaken to ensure that some expertise is built up until SOA is deemed a safe choice. The wait and see approach has obvious competitive implications: Full scale SOA adoption is a multi-year program even for organizations with established governance culture. Many organizations taking the *wait and see* approach are less mature and hence will have little opportunity to catch up when competitors begin to leverage SOA effectively.

# Open Source Software in the Enterprise

## Open Source Myths

Before exploring how Open Source Software can address the problems facing anybody building the business case for a SOA initiative highlighted in the previous section, it is useful to review how Open Source Software is currently used in the enterprise and for what. It is fair to say that Open Source Software (OSS) has come a long way over the last few years, with offerings such as Linux, Mozilla Firefox, and the JBoss J2EE application server becoming mainstays in many IT departments. What is surprising is that for such a significant trend there remain many myths around Open Source. These may add to the romantic appeal but can actually hinder OSS adoption in the enterprise as the decision makers may be put off by the myths.

### The 1st Myth: OSS as altruism

The first myth, still promoted by many industry observers and OSS promoters, is that Open Source is an exclusively altruistic endeavor (sharing your intellectual property and expecting others to return the favor) with a vast network of individual developers working in their free-time for the betterment of all. A more sophisticated version of this is that OSS development is driven exclusively by customers, the end-user organizations who see benefit in sharing the cost of developing the Open Source software with other organizations who have similar requirements. The first version of the myth was true to a degree with Linux early on. However, Linux is actually an atypical example of the vast majority of OSS projects as it implemented a very mature and widely understood technology with a pre-existing and huge community of developers and users who knew UNIX™ inside out. The customer driven OSS myth was always mostly a myth: On rare occasions, end-user organizations have either open sourced an internal development or funded a new OSS project in order to share on-going costs. However, these projects have almost always involved an OSS vendor to drive the development of the project beyond the initial open sourcing. The reality is that vendors from industry giants such as IBM to integration software specialists invest significantly in the OSS movement and reap considerable revenues from this investment. This is actually a benefit to adopters of OSS as the software is being developed by specialist developers who are writing the Open Source code as part of their paid employment.

### The 2nd Myth: OSS is free

The second myth is that OSS is free. This equating of zero license fee with free may apply to personal use but does not apply to commercial deployments. Even software with no license fee requires investment in skills, in the actual development of the solution and in maintenance of the resulting deployed solution.

## Enterprise OSS: You are already using it

A reasonable question to ask is how widespread is OSS adoption in the enterprise? Unlike closed source software where vendors are typically public corporations required to disclose revenue information, the OSS free distribution model makes it much harder to accurately estimate the extent of its use. Further complicating the measurement is the habit among the OSS community to use download statistics as proof of adoption. The often huge numbers claimed inevitably include a high proportion of repeat downloads, downloads for evaluation purposes, downloads by developers driven by simple curiosity as well as by non-commercial users.

Unfortunately, the use of these download numbers actually clouds and undermines the real success of the projects. What is clear is that OSS is being used successfully in many organizations: Most organizations have at least explored migrating some aspects of their infrastructure to Linux. While the precise level of usage remains the subject for heated debate, it is also clear the open source Apache Web Server remains popular for corporate web-sites. In the integration software area, Red Hat's JBoss J2EE Application Server is among the top 4 choices with IBM, Oracle and BEA. On the development side, the Eclipse project, originally developed by IBM and then open sourced, has become the

Integrated Development Environment (IDE) of choice for Java developers.

These examples only reflect the visible usage of OSS in the enterprise as the closed source vendors are also keen adopters of Open Source Software within their own products which are in turn deployed into the enterprise. In the SOA space, key OSS components such as XML parsers and XSLT engines for message transformation are widely embedded within closed source products. Reflecting the already mentioned popularity of the eclipse IDE, the closed source SOA vendors also often supply plug-ins to this IDE to allow developers to use their preferred development environment.

Finally, there is another route by which OSS is deployed in *stealth* mode: Development teams recognize the value of an OSS project to address a specific problem. The software is downloaded, developed with and deployed; all without the need to engage with the normal procurement and legal processes. This approach has a significant and often under-estimated implication on IT risk which will be explored later in this paper.

# SOA and OSS

## Addressing the SOA budget challenge with OSS

The historic problem with justifying the business case for integration projects has been that integration is expensive and is seen as a cost as the business benefits are buried within subsequent business-focused projects. Furthermore, integration projects are often shared between multiple departments that they link. This means that the budget must also be shared which is often hard to achieve organizationally.

The EAI-style broker-based integration solutions, popular until the emergence of SOA, often had a substantial up-front infrastructure cost. This is because the broker-based architecture is of the 'hub-and-spoke' type, where all flow between components has to pass through a central hub. Therefore, even if the need is only to join two applications, the same hub is required as would be needed to join thousands. The cost of this hub is often a major inhibitor to initial integration projects, frequently running into hundreds of thousands of dollars.

As was discussed in the section "SOA and the Budget Catch 22", this core issue with funding remains with SOA projects. Even though modern approaches to integration, such as the Enterprise Service Bus (ESB), offer a cheaper approach, licenses will still be required at each node to be integrated. For instance, an ESB license may only cost around $10-50K each but this still represents a substantial outlay. When combined with other SOA infrastructure such as a registry, and various other tools, the project cost can be daunting. Furthermore, the scale of this outlay makes it difficult to build a business case when the tangible benefits may be hard to measure, the total cost will be hard to estimate over the lifetime of the SOA deployment and charge-backs or central funding models are hard to implement.

The way OSS addresses this is both obvious and subtle. The lack of upfront license cost is an obvious attraction even if as most IT professionals realize software license costs make up a small component of the overall cost. However, even if the additional support and maintenance costs are factored in, the overall cost is still likely to be much less than going with a commercial software vendor's solution. Furthermore, the lack of software license cost and the reliance on generic contracts (such as the famous General Purpose License, GPL) means that the procurement process is either greatly simplified or sometimes bypassed entirely. All of this reduces the cost implication associated with completing a project using SOA.

Of course, it should be stressed that this assumes that the project is capable of satisfying the project requirements. This apparently obvious statement becomes particularly significant if the OSS project is promoted by a vendor as a pathway towards a much more expensive closed source product. In this case, OSS functionality may be deliberately limited for commercial reasons in order to encourage transition to the *fully featured* product. This type of OSS solution should not necessarily be excluded as there are still potential cost benefits: For instance using the OSS version to kick-the-tyres before transitioning to the full strength version. Alternatively, the OSS version may be used in conjunction with the closed source product to handle simpler problems within some parts of

a widely deployed solutions. However if the intention is not to transition to the closed source product, additional vigilance is required: When considering one from this category of OSS solutions, it is essential that there is clarity around what the OSS offering is suitable for and whether the project will extend beyond this.

Returning to the way OSS in general addresses the SOA budgeting issue, OSS projects can also reduce the impact of the second and third issues listed in the box titled "*The SOA deployment risks*": Predicting the requirements over the life-time of the project and hence estimating the total cost of the SOA initiative. Open Source Software projects intended for use in SOA tend to work in terms of frameworks, where the basic skeleton is implemented together with plug-ins to add functions when necessary. The lack of additional upfront investment and associated procurement activity means that new functions can be tried out with minimal effort. This incremental model for investment is attractive from a business case perspective as it allows the project to be realistically broken down into stages. Each stage can be completed with confidence that it will be easy to move onto the next phase and if necessary expand the functionality set required. Contrast this with commercial software where required procurement processes are time-consuming and software vendors bundle functionality to justify a higher price, resulting in companies often buying a full stack of software regardless of how much functionality is actually required at the time. This drives higher up-front investment, and often leaves products on the shelf, causing problems for future product acquisitions. With OSS software, it is easier to simply take on the software needed at the time, and then add to this as and when required.

## *Using OSS to address SOA risks*

There remains a perception among some analysts and IT professionals that the risk associated with Open Source adoption in general is much higher than that associated with closed source software. This reflects a lack of experience with evaluating risk related to Open Source adoption rather than necessarily higher risks. This is not surprising: end-user organizations have learnt how to evaluate the risk associated with closed source products over many years of sometimes painful experience. In fact, a careful analysis of the risk of OSS adoption for SOA shows that the risk profile is clearly different. Furthermore, the overall risk can be lowered significantly when OSS is adopted in partnership with a successful OSS vendor.

The first area of risk highlighted in the table on page 6 and in the side bar, is organizational maturity. This is outside the scope of any software package to address—although software may enable and facilitate the development of the required maturity.

The second area highlighted relates to the lifespan and criticality of the deployment. The reality is that integration software frequently plays a mission-critical role. As more applications are integrated together, the integration software forms the nervous system for the enterprise, and any failure in this nervous system will have serious impact. At first glance it may appear counter-intuitive to suggest that Open Source actually lower this risk from this perspective. To investigate the claim it is necessary to consider the main areas that are assessed when measuring the risk associated with a closed source vendor.

| Areas of closed source software risk | Explanation |
|---|---|
| Vendor risk | This risk will vary with the size and success of the vendor. The risks are that the vendor may disappear through acquisition or liquidation or that the vendor strategy may change so radically as to undermine the viability of the current products. |
| Product risk | Product risks are associated with either escalating costs and requirements to maintain specialized skills limited to this specific vendor and the risk that the vendor will moth-ball the product |
| Feature risk | The risk that a specific feature or customization is not maintained as part of the standard product. The customer will then be exposed to an additional vendor imposed cost associated with the maintenance of this feature indefinitely. |

### *Recall: The SOA deployment risks*

The key areas to be considered are:

1. A need for organizational maturity
2. Long life-spans and business criticality of deployment.
3. A high degree of customization to support business requirements.
4. A need to support business changes throughout the lifespan of the deployment.
5. The availability of skills and expertise required to be successful

If for some reason the vendor, product or even feature is no longer available, this has potentially severe cost implications for the organization and may even impact on its ability to sustain the solution at all. Open Source addresses this risk in two related ways. Both give the end-users much greater control over the software they have deployed than is the case with closed source software:

The availability of the source: For larger companies with substantial technical resources, having the source of the integration software might be very attractive. For these types of companies, support can probably be provided in a more immediate fashion by using internal resources, but this is only possible if the source is available. It provides an opportunity to maintain features and drive functionality required by the organization independently of any vendor.

The existence of a user community: Most OSS projects will have an associated community of developers across the world who may be involved in improving and expanding the project, at least reporting bugs and exchanging knowledge on the project's use. The philosophy of open source is that whenever someone makes changes to improve or enhance the particular OSS product, these changes will be made available to the rest of the community, potentially building a constantly updated and evolving offering. When this works, the benefits are clear in terms of shared cost of new developments and reduced risk of failure: The community provides an alternative route to support, maintain and develop the software overall and at a feature level independently of any vendor who may have been involved in the project. As a shared endeavor, it allows organizations unwilling to take on the cost and risk associated with exclusive development to participate.

When considering the need to facilitate the high level of customization required, the availability of the source code also plays a role. In addition, Open Source Software projects intended for use in SOA tend to work in terms of frameworks and make heavy use of standards. All of this can deliver a high degree of flexibility and adaptability. Users can add developments to the source and make local changes and they can more easily plug the OSS offering into other technologies from messaging queuing systems to core applications where required. These same benefits apply equally to the need to evolve the solution inline with business evolution.

Finally, OSS projects partially address the issue around the availability of skills. The solution is only partial for two reasons:

- SOA initiatives also require business and organizational skills which are outside the scope of any software package whether open source or not.
- SOA OSS projects require developers with very specialized skill-sets. It is important to distinguish between the skills required to develop the software and the skills required to use the software, whether Open Source or closed source. Most organizations will only require and pay for staff capable of using the software to solve business problems directly. They will be less willing to retain the more specialist and expensive staff capable of developing such software. This leaves a potential gap in OSS projects which Is typically filled by the OSS vendors.

However from the perspective of acquiring those skills required to use the software, OSS projects have significant benefits as they are typically heavily reliant on standards and will use generic development tools and methodologies. This means that it is easier to find developers with the appropriate skills with the project specific communities acting as mechanisms for accessing the pools of project specific expertise.

## Why SOA suits OSS

An analysis of most OSS projects that have achieved sustained enterprise momentum highlights two broad characteristics which may initially suggest that OSS and SOA are not a good match:

- Low-risk OSS where the solution is well-understood or has a low cost of switching out to a closed product if OSS proves unsuitable. (E.g. Mozilla Firefox) or
- OSS which provides commoditized functionality where there is little scope for vendor differentiation. (E.g. Linux).

### OSS Communities

Open source software projects are based on the principle that the software is available to all, and that typically anyone can contribute to it. This principle and the *safe environment* created by the Open Source licenses such as GPL generates the concept of communities of developers, all interested to varying degrees in the code base, its health and its on-going vitality.

Often, OSS communities take considerable pride and ownership, and community members are frequently not only users but also strong advocates for the software.

Clearly, integration does not qualify on either count: It is of its essence critical to the business and it Is not commoditized as there are still many different approaches to solving integration challenges. That OSS is today being used successfully for integration projects and SOA projects forces the conclusion that the problems being addressed are fundamentally different. The differences are in fact straight forward and hence the reasons for OSS success can be easily discerned:

**Most OSS is written by developers for developers and SOA requires technical solutions:** OSS is dominated by the developer community and tends to be weaker on other aspects of the development process found in commercial closed source development. In particular, areas of tooling to allow less skilled users to utilize the software and to a less degree documentation are common weaknesses. This has limited the adoption of OSS outside domains with very commoditized tools (such as email or web browsing) or to more technical areas where the user is a developer. SOA is an example of the second domain. Furthermore, the wide adoption of Eclipse among the Java developer community has reduced the need for projects to create their own tooling as eclipse provides a well-known development environment into which project specific tooling can fit. Documentation is also less significant because of OSS' active user communities and ethos of mutual self-help. Therefore, for integration in general and SOA in particular, these OSS drawbacks are not major obstacles.

**Integration remains a highly technical area with business knowledge as secondary:** OSS projects are typically focused on solving technical problems. For this reason, business oriented OSS projects, for instance those providing ERP functionality, have not had much success to date. In contrast, integration remains an area where technical skills are primary. While great progress has been made over the last 10 years in making it easier to integrate across technology stacks, the challenges have simply moved up the stack to address more sophisticated technical problems such as data semantics. Therefore, the technical focus of OSS is well suited to integration.

**SOA OSS projects are innovating:** The adoption of OSS has often been associated with cost reduction strategies such as consolidation or virtualization based on Linux. In such situations, OSS projects are focused on freeing existing solutions from mature and well understood vendor owned Intellectual Property constraints. Integration and in particular SOA is at very different stage of its evolution. In the SOA domain, OSS is being used as the basis of innovation. However, OSS driven innovation is different from closed source innovation at one level: the commitment to interoperability and openness. It is notable that OSS SOA projects tend to be much more keen on key interoperability standards, such as the Java Business Integration (JBI) standard, which makes it easier to plug together components from multiple integration stacks.

# *OSS: The risks and pitfalls*

Over the last twenty years, most organizations have become comfortable with assessing the risk associated with the adoption of closed source products. This paper has shown how Open Source addresses significant risks associated with closed source software. However, it would be naive to claim that OSS adoption is without risk. As well as the risk that the project will fail—just as a closed source product may fail—there are risks associated with weak version control, incompatibilities between OSS projects and lack of deep project expertise. However, careful analysis suggests that when adopted in conjunction with a commercially successful OSS vendor these risks can be mitigated to a large degree.

## *Risk analysis*

One of the inhibitors to systematic adoption of OSS beyond the most established OSS projects such as Linux has been the concern about risk inherent in the OSS model. While this concern originates with the caricature of the anti-establishment OSS developer, it is reinforced by the perceived lack of a *"throat to choke"* and legitimate concerns about the re-skilling costs associated with any major technology adoption.

At the project level, experience of incompatible and ever changing versions and the vast array of Open Source projects have provided further grounds for avoiding OSS. This has meant that where OSS is adopted it is often driven from the grassroots, with developers

finding success with OSS and only then promoting formal adoption within IT management. Typical OSS risks are highlighted and analysed in the context of SOA below:

| Type of risk | Severity | Explanation |
|---|---|---|
| **Hidden Intellectual Property:** The Open Source Software may include third party IP and expose the end-user to legal claims. | **Low** | Experience shows that it is highly unlikely that such IP exists and even more unlikely that the owner of the IP would successfully pursue the end-user. It is much more likely that the offending project would be modified to remove the breach or that the vendor would reach an accommodation to ensure the continuance of the project. |
| **Security issues:** the Open Source Software may includes security holes which a knowledgeable hacker could use to attack the organization. | **Low** | This is clearly an issue for security specific OSS projects such as firewalls. However in this area the consensus is that making the source open is actually a more effective route to ensuring security as many more eyes will examine it for faults when compared to closed source software. In the integration and SOA area, there is less scope for such issues to occur and likely to be much less interest from the hacker community who naturally focus on security specific software. |
| **Governance:** Open Source Software can be downloaded, developed with and deployed outside of normal IT policies. | **Medium** | This happens when governance relies on procurement and legal review of contracts as the mechanisms for ensuring that policies are followed. The implications of this are additional costs associated with incompatibilities; costs associated with retaining skills required to maintain the software and the risk associated with not knowing what software is in use. |
| **Integration cost and Version control** | **High unless mitigated by an OSS vendor.** | Most reputable closed source vendors provide guarantees around interoperability of different versions or software components. If OSS projects are simply downloaded, there is a significant risk of version mismatch. While careful test in the end-user organization will alleviate this problem, the cost and risk must be dealt with each time versions are upgraded. |
| **Skill costs:** Unavailability or high cost of skills required to maintain, develop and patch software. | **Medium without service agreements with OSS vendor.** | While it may appear an attractive option for the organization's developers to patch software themselves, this requires specialist expertise to be maintained inside the organization. The community may provide patching capabilities but there is no guarantee that patching will occur within acceptable timeframes. |
| **Continuing innovation and development within the project** | **Depends on the scale of the community and/or commitment of an OSS vendor** | If an OSS project does not have an active community, the risk of project failure is clearly high. Popular projects will maintain momentum but will typically rely on a small group of key highly skilled and valuable individuals. If these individuals work for another end-user organization, there is a medium risk that they will be redeployed to another project. If a non-vendor sponsored OSS project does not have an active community, this will seriously undermine the viability of the project overall. |

Underpinning all of these concerns is the fact that Open Source software is, by definition, distributed to all under the OSS licenses and open to all to contribute. The implication of this is that OSS is presented on an "as is" basis. While closed source software license agreements are often also "as is", there is a level of control and assumed responsibility which is lacking with OSS. Without the presence of a vendor, the only support for the software comes from the community that is working with it. While this frees customers from some of the vendor risk, it does expose them to the risk that the community will not provide services to alleviate these risks. In the final analysis, these services can only come from a business specialising in the selected project or projects. It is in addressing these risk areas that the OSS vendor's value proposition lies.

# Evaluating SOA OSS vendors

OSS can be viewed as an attractive option for SOA because it addresses key problems faced with initiating SOA programmes. However, it has also been shown that the OSS model is not without its own drawbacks. It is to address these drawbacks that OSS vendors have emerged. The organizations in this category can include businesses focused exclusively on promoting a specific set of OSS projects, businesses deriving revenue from both OSS and closed source software and businesses focused on general consulting services as well as OSS specific ones.

An evaluation of an OSS vendor must start with the same aspects as would be followed with any supplier: How viable is the vendor? Will it make a good business partner over the long life-span of the deployed system? Are its business motivations inline with the customers goals? Is it committed to the business and will it provide good services? However, OSS vendors are not simply providing support, maintenance and consulting services, as the projects supported and the value delivered varies greatly. Therefore, the evaluation must encompass vendor analysis, product analysis and service analysis. As such, it means that OSS vendor analysis is much closer to closed source vendors than the pure consulting companies they are sometimes compared to.

## Evaluating OSS vendor's strategies

Beyond a general evaluation of the OSS vendor as a potential long term partner, there are five specific areas where they may add value that should be considered:

### Integration and version control

For closed source software products, this is a core part of the development process paid for with license fees. For that reason it is often overlooked when considering an OSS offering. However, anybody with experience of piecing together a complex set of integration software components into a usable whole will understand the level of frustration, technical difficulty and cost associated with the task. Furthermore, the impact of a failure to integrate the infrastructure is typically fatal for the project. For this reason most OSS vendors provide an integrated set of supported OSS projects which they certify will work together.

A side effect of the open source development model is that versions will appear more regularly than with closed source. When attempting to use OSS for SOA, it is highly likely that the software will come from multiple projects. It is also probable that the OSS will have to integrate with third party software (either infrastructure such as MQSeries or applications such as SAP). Unfortunately, this increases the risk that the interplay between projects and versions will result in unexpected incompatibilities appearing as new versions are selected and deployed. Addressing this risk requires extensive and costly integration testing or fire-fighting when an incompatibility is discovered. To quantify this with a simple calculation that shows how many combinations must be tested:

***5 OSS projects x 3 current versions per OSS project x 3 other pieces of software (each with potentially 2 versions active) = 45 permutations***

This, sometimes called the *"matrix of death"*, is not a problem with OSS only: Closed source software vendors must also complete such integration testing prior to release. However it is an expensive problem and represents up to 20% of the total development cost for closed source integration software vendors. Therefore, it is clear that any end-user organization should think hard before deciding to take this risk and cost on themselves. For this reason a key part of many OSS vendor value proposition is the provision of a certified set of required OSS projects guaranteed to function correctly as a bundle.

### Innovation: Technology

As was explained earlier, Open Source's origins are in commoditization of well established software—there is no better example than Linux itself which is an IP-free recreation of UNIX. However, this is not always the case. Areas where there are specific reasons to innovate with OSS have followed a different path—for instance in firewalls and security. Integration is also following a similar path with projects sometimes breaking new ground

and being truly innovative. Some vendors have decided to focus on this approach—typically developing significant intellectual property which is made available as OSS. Such specialization is driven by a different analysis from the mass of OSS projects. This may be a focus on data mediation or reliable messaging. By focusing and driving innovation in these areas, the vendors develops unique expertise and also control to ensure the project will meet its customers' needs.

### *Innovation: Usability and productivity*

This area of innovation is highlighted because it is not an area typically associated with OSS projects. However, it is a key requirement for SOA because SOA by its nature democratizes integration: For SOA to be effective, technical developers in business units are now expected to participate in SOA related project where before integration specialists would have been used. This puts an additional pressure on skills which is being addressed by some OSS SOA vendors. Vendors are taking several different approaches: Either to create Open Source GUI tooling that would previously have only been expected in closed source products. Or to develop frameworks that both guide the developer and provide pre-canned components for common integration challenges. While it should be stressed that neither approach makes it possible for business analysts to use the tools, it does make it easier for the less technically skilled developers and increases the productivity of all developers.

### *Community*

While the idea of altruistic mass participation in most OSS projects is a myth, this does not mean that the secondary innovation of OSS, organizing communities to create software, is not valid. Communities direct new development towards real customer problems and act as a clearing house for customers to exchange solutions that they have built themselves. The power of a community is only as strong as the breadth and depth of the expertise available to address the problem. In the case of OSS projects related to SOA, some areas are technically difficult and require very specialist skills. These skills may not be available in the end-user community as organizations may be unwilling to devote these expensive and unusual skills to the project long-term. In contrast, OSS vendors are motivated to develop such skills and use them to sustain the project.

However, not all aspects of an OSS SOA project will be so complex. There are others where the community can contribute the expertise to solve specific community friendly developments such as application connectors. The example of application connectors demonstrates how communities can develop high value components: Lustratus estimates that between 50-75% of integration project costs relate to building the necessary adapters, frequently done by getting a third-party software house to do the work. These components rely on the skills typically found in end-user organizations while embodying little intrinsic value that the developing organization would want to own exclusively. In this mode, the OSS vendor becomes the community facilitator as well as taking on the role of tester to ensure that the contributed source is compatible across the project.

### *Skills: Support and consultancy*

OSS vendors act as centres of excellence for key contributors to projects. These contributors can focus on the technically challenging areas which do not directly interest the end-user developers. They can also act as community organizers and facilitators. In both cases, it is unlikely that end-user organizations can justify retaining these individuals on a long term basis. However, they can be vital during all phases of the SOA project. Early on the expert developers can provide insight into the technical details of the project and the facilitators can provide guidance on how to work with the community and the likely direction of the project. Later in the process, design and development expertise can be tapped to assist or accelerate completion of the project or to troubleshoot.

# *Conclusions*

The decision to deploy any new software requires careful analysis of the benefits, risks and costs associated with potentially relying on that software for many years to come. Organizations have sometimes been reluctant to adopt Open Source Software due a lack of understanding and experience of evaluating OSS projects and vendors. This paper has addressed that lack of understanding by analysing why OSS makes sense for

# *Appendix: A short history of Integration and SOA*

Connecting together the ever increasing network of applications, departments, clients and suppliers is an ever increasing IT challenge. The business drivers for integration appear, in many cases, overwhelming – more efficient, streamlined processes, faster execution and reaction to market changes, improved leverage of existing IT investments and assets, a reduced IT cost base built on shared services, all contributing to mitigation of business risk.

While the scale and importance of the problem has increased over the years, businesses have since the earliest days of networking had to deal with the issue of getting different applications to interact. For many years, this was achieved through primitive mechanisms such as file transfer, or hard-coded communications programming. This tactical approach to integration solved the problem at hand and is still relied on in many cases. Their tactical nature meant that each solution was built from scratch. This resulted in escalating costs to maintain what rapidly became a rat's nest of point to point solutions. Furthermore, the technical limitations meant it was hard to achieve real-time integration and easy to introduce inconsistencies in business process or security model.

In the early 1990s, software began to emerge that was designed specifically to address the needs of communication between different technology environments designed to support both real-time integration and support enterprise qualities of service. Perhaps the most successful of these initiatives was message-oriented middleware (MOM), providing a usually asynchronous method of passing messages between applications and platforms. The Enterprise Application Integration (EAI) market was born.

However, the need for integrating applications continued to grow exponentially, driven by business needs such as handling mergers and acquisitions, streamlining processes and increasing efficiency and customer responsiveness. The basic MOM communications layer was extended with message brokers to provide more benefits at the application level, such as delivering added-value like data format transformations, intelligent rules-based routing and pre-built application adapters. But these proprietary EAI stacks proved expensive and required a considerable pool of specialist skills to deliver real value.

Attempting to address these issues, web services, a standards-based way to describe and integrate independently developed pieces of code emerged. These standards leveraged the internet standards (HTTP and XML) and distributed computing model such as CORBA to reinvent EAI as an independent and light weight model of application integration. In parallel to the development of Web Services, Service Oriented Architecture (SOA) emerged as a general architecture for application integration which recognized that application integration is much more than simply a technical problem.

The core concept of SOA is the breaking down of IT applications and systems into a collection of 'services' that can be invoked through a standard interface, with no knowledge needed of the location and execution point of that service. This service based approach yields the opportunity for developing a shared pool of reusable business services. By reusing existing services rather than building new one reduces the total set of IT assets and hence reduces maintenance costs. By reducing the amount of new code developed on each project, there is a faster time-to-market and improved return on assets. The use of a clean interface to invoke the service tied with the isolation of the caller from knowledge of the service implementation makes these services usable from any environment.

One of the conceptual challenges at the heart of SOA is the balance between architectural approach and the need for appropriate technology to implement it with. By taking an architectural approach which operates at the level of business service rather than function, it is easier to align the business and IT perspective. A major effect of this change of emphasis is the increased focus on key challenges of rolling out SOA across any enterprise: governance and the behaviour changes required to be successful with SOA.

While it is possible to build a SOA without introducing new technology, SOA projects will in most cases involve solving technical problems which cannot be simply solved through architecture. Therefore in almost every case, SOA programs involve the introduction of modern technology which enables the new architecture while working in conjunction with existing IT investment.

While the precise definition of what is needed in SOA enabling software product is still evolving, the general properties of the software ecosystem is clear and covered in the **Lustratus Insight**: *"The need for a SOA ecosystem"*. One of the most popular products used to implement SOA is the Enterprise Service Bus (ESB). These provide a lighter-weight, standards-based integration stack as an alternative or an adjunct to EAI broker-based platforms. Web services standards are supported by ESBs, and in addition to Message Orient Middleware functionality and basic mediation functions like routing and transformation. In essence, ESBs provide a 'good enough' way to handle the integration needs inherent in the SOA concept.

## *The organizational challenges of SOA*

A primary benefit of SOA is reuse of services between projects. Good service design while important is not enough to achieve this. Also required are significant organization changes:

**Governance** structures and processes must be established to involve stakeholders in the definition, evolution and compliance with best practices and processes.

**Behaviours** among IT and business staff must be changed to encourage greater communication and reuse. This change can be achieved through changes in role definition and incentive structures.

## *What is an ESB?*

Enterprise Service Buses (ESBs) offer a standards-based, general purpose approach to integration needs. Typically ESBs provide

- Support for standards such as web services and XML
- Asynchronous and synchronous connectivity options
- On and off ramps for common application environments
- Transformation and intelligent routing services


Lustratus Research Limited
St David's. 5 Elsfield Way, Oxford OX2 8EW, UK
Tel: +44 (0)1865 559040

www.lustratus.com

Ref RB/LR/56300411/V1.0