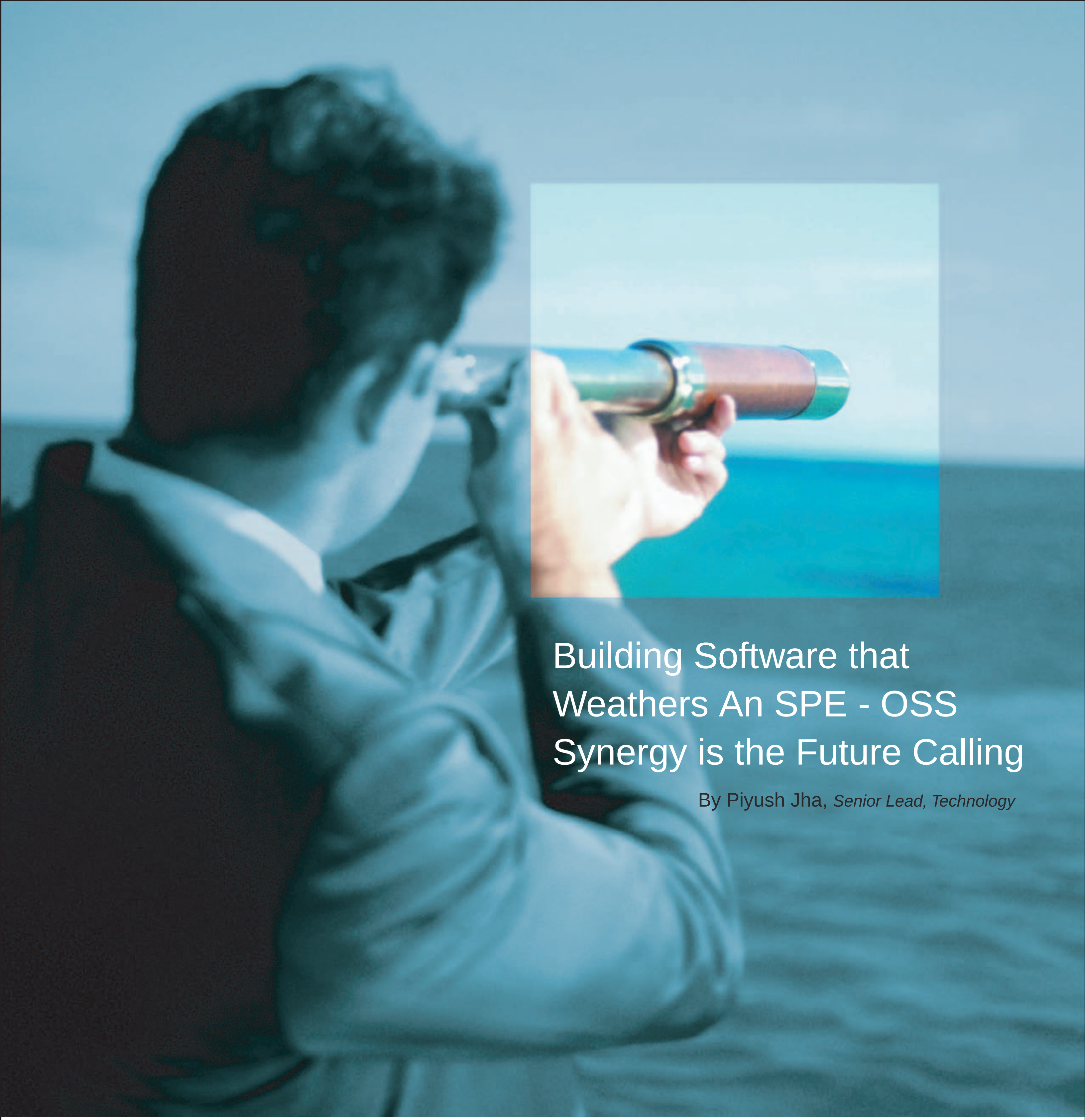

Building Software that Weathers An SPE - OSS Synergy is the Future Calling

By Piyush Jha, *Senior Lead, Technology*

Index

Executive Summary

The software topology has changed drastically over the last 50 years and so has the kinds of problem software is used to solve. Today, software solves all kinds of mission critical problems, so we need to change the perspective of "just make it work" - which is sadly the way most software is written today. Software Product Engineering is a fast emerging concept pioneered by VAssure which believes in the paradigm that Enterprise Applications require a higher level of quality, robustness and flexible architecture. It is worthwhile to design and architect software in a manner that the underlying framework weathers the test of time and remains unaffected.

Frameworks provided by the OSS community like the Hibernate or Wiki or Apache are more usable and reliable than their propriety counterparts such as Entity Beans, SharePoint or Microsoft IIS. The only drawback is that all OSS tools are not business process ready and need to be carefully evaluated and enhanced before induction. This is where the SPE - OSS synergy can be of great use. While SPE as followed at VAssure believes in leveraging the open source advantage as much as possible, its robust architecture, evaluation and delivery mechanism ensure that the released product is fool proof and complete.

We have benefited using this approach on quite a few of our projects and yet again, we feel we are pioneering (and formalizing) a paradigm that would change the way software is written today. This paper makes a case of what the drivers are and the changes that are required. We should follow it up with a series of papers on how to implement the same.

Change in the Software Topology

Not long ago, computers were for scientists and military only, laboriously programmed by changing connections with soldering irons or switches. Today, software has touched almost all aspects of human life and is trusted to solve all problems. But then, "with great power, comes great responsibility". One wonders, whether the software industry of today is mature enough to handle such responsibility.

Kinds of problem software solves today

The software of today solves all kinds of problems - mission critical, life saving, disaster management and real time; and the human society depends on these. One bug properly placed and innocuous might have the capacity of destroying the whole mankind (imagine one in the US nuclear reactor to set it off on the 1st January 2007).

Also, the industry is here to stay, like any other industry that makes the bridges and dams possible, which makes the clothes we wear possible and which would one day establish human habitat on the other planets.

The reliability puzzle

Gerald P. Weinberg once famously observed that, "If builders built houses the way programmers built programs, the first woodpecker to come along would destroy civilization." He was right. Up to now, the reliability of most software has been atrociously bad. On of the latest versions of Windows from Microsoft - the most popular consumer level Operating System - was acknowledged to have an estimated 63,000 minor bugs in it.

http://www.cnn.com/2000/TECH/computing/02/17/windows.2000/

Software that would last 200 years

Many things in society are long term; in comparison software still is relatively short term. We do not build bridges with the assumption that a bridge1.0 will be replaced by bridge1.1 in two months and then bridge2.0 in a year. There is nothing wrong with our capacity and the flexibility to be iterative, but software if built with this assumption has more bugs and a weaker base than it should have. The society today is ready for software that would last, and not infected by the "ongoing business entity" and "will fix in the next version" mentality.

In the early days of computer science, the software was intimately connected to the hardware on which it ran, and as that hardware was replaced by new, better hardware, new software was built to go with it. Today, hardware is capable enough that software can be written that will continue to run unmodified as hardware is changed. While being able to add features is an asset, the robustness of the basic framework should be beyond doubts.

Dan Bricklin in one of his papers has emphasized that this industry should also learn the rules of some other time-tested industry. For example, the software industry could learn the following from the civil industry:

- For years we emphasized execution speed, memory constraints, data organization, flashy graphics, and algorithms for accomplishing this all. Curricula need to also emphasize robustness, testing, maintainability, ease of replacement, security, and verifiability.
- Standards bodies publish best practices. Like all engineering, new software, as we know, commits old errors.
- The role of independent testing entities should be emphasized and raised.
- When projects fail, public enquiries should be performed.

Enter Software Product Engineering

Software Product Engineering is a fast emerging concept which focuses on alleviating many of these ills and adds features to software development which were not stressed before. It is fast changing the way the producers and the consumers alike looked at software development.

SPE Definition

Enterprise software products require a higher level of quality, robustness and flexible architecture. Software Product Engineering works on attending to these needs of the applications. While building such applications, the robustness and reliability of the architecture are paid sufficient attention to — even if new features are added and the interface of the product needs a revamp, the underlying essential elements remain the same.

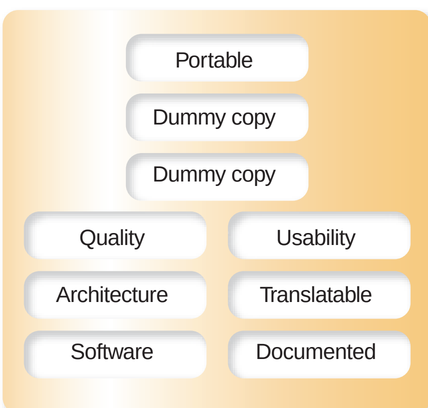

Figure 1 The cube above captures the essential differences that we at VAssure believe, are important, when developing and maintaining enterprise grade solutions.
More information on this can be found at the VAssure website, the pioneers of this technology and the global leaders of the SPE space.

Solving the Reliability and Robustness Puzzle

As discussed in the last section, where most software development fails today is the ability to deliver only the products which are well tested and reliable and do not fail at critical junctures. This requires a complete cycle of analysis, architecture and design, peer reviews, iterations and phased releases which ensure that the product meets the quality requirements specified.

Following this complete process gives the ability to build on existing partial solutions or extend products at various stages of its lifecycle and yet maintain the same standards of quality. VAssure teams have built products for Cingular and Polaris where even for a complete overhaul of the presentation, the underlying framework underwent none or minimal changes. We shall discuss further about the importance of each of these steps and the integrated process in a greater detail in the sections to come.

> *"It's a waste of energy to try to stop something with a life of its own. The market will always win."*
>
> *...Doug Heintzman, director of technical strategy for IBM software*

Value Assured

The Open Source Software Paradigm

The Open Source initiative has amazed and attracted a majority of sharp and intellectual individuals of our times. The idea of collaborative problem solving is so interesting and value additive that some authorities on the subject feel there is no other way software development can go. Open source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code.

The OSS Definition

Very briefly, OSS/FS programs are programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program (without having to pay royalties to previous developers). Please go through The Open Source Definition for a complete definition.

The OSS Edge

> *"Ten years from now. we will look back and say, 'What did we do before open source?"*
>
> *...Marten Mickos, CEO, MySQL*

The Open Source paradigm and has shown great success with projects like Linux, the Apache web server software, MySQL, Hibernate, maven among others. In developing an open source solution, the need is first recognized by a person who requires a solution to this need, and he sets about making a solution on his own, for his own use - but he also releases it to the public at large. From there, others may add to it, embellishing it more and more until it is robust enough to handle most people's needs. At every iteration of this development cycle, communication with the other developers and users is crucial and forms a beneficial feed-back loop. The project increases in complexity but so does the number of people working on it and critiquing it. The result is usually a relatively bug-free and robust result delivering reliability and capability.

OSS is Inevitable

The success stories of Linux, Apache, Hibernate and many others of the paradigm go on to prove beyond doubt that this is a path worth treading. The simple sense of power and collaboration that it gives is enough to solve the toughest problems.

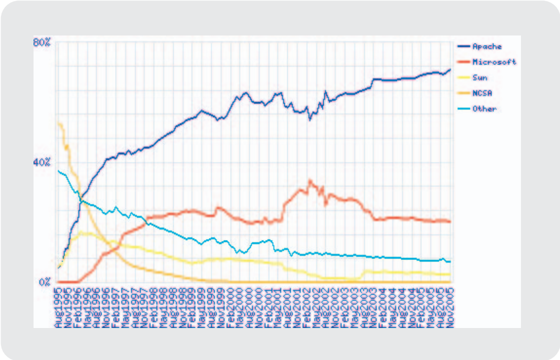

Figure 2 Apache as compared to other web servers

In the early 1980s people were saying, "The free software people build some nice toys and demos, but they haven't got what it takes to build real tools." The FSF proved them wrong. Before 1991 the same people said "OK, GNU is a nifty programmer's toolkit but they'll never build a viable operating system." Linux proved them wrong again. Now they're saying "OK, so Linux is a nice sandbox for hackers and it does Internet pretty well, but they'll never build decent end-user applications." If the naysayers are right this time, it will be a first.

Even in terms of security and reliability, OSS tools rank far ahead. Closed sources do three bad things.

1. They create a false sense of security.
2. They mean that the good guys will not find holes and fix them.
3. They make it harder to distribute trustworthy fixes when a hole is revealed.

Open-source operating systems and applications are generally much more security-safe than their closed-source counterparts. When the "Ping o' Death" exploit was revealed in 1997 (for example) Linux had fix patches within hours. Closed-source OS's didn't plug the hole for months. Alan Cox has written an excellent article on The Risks of Closed Source Computing

OSS Myths

One common myth about using OSS is that its main selling point is that it is free. In reality most of the OSS today is used because of its reliability, flexibility and the ease to use it. Another myth is that OSS can be used at will or if one uses OSS, the code will require to be released. The truth is somewhere in between these, it actually depends on the license one is using OSS under.

Also, there are many products like confluence, JIRA and others which add value to an open source solution like wiki and sell it at a very low cost as compared to their propriety counterparts. This is made possible only because the existing infrastructure they get to build upon. So, using OSS is definitely a good source for revenue.

OSS Risk Mitigation

Despite mentioned benefits of Open Source Software, they are not free of ills. This is especially true if they are to be used not as a tool but as a part of a deployable mission critical application. This paper proposes that before open source software is used for such an application, they should be properly evaluated for their Business Process Readiness. Also, the license of the particular software should be checked so that no condition is overlooked and infringed due to negligence.

Business Process Readiness

Before using open source software as one of the components of an application, a proper evaluation requires to be made about the readiness of the software to be used as a part of a professional business application. This model was first suggested in a paper at www.openbrr.com. The assessment is based on the kind of role the software is going to perform and the maturity of the entire product which is going to use it. It uses a phased approach for a faster analysis.

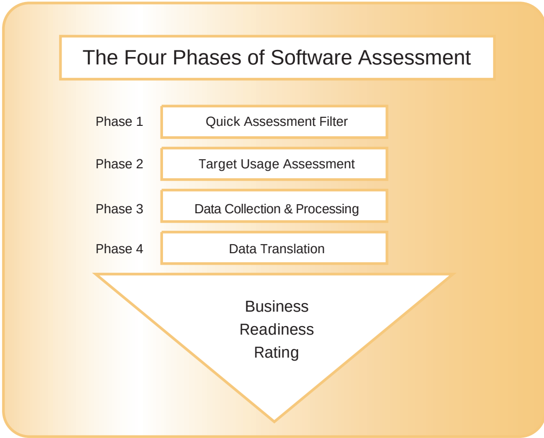

Figure 3 Phases of Assessment for Readiness.

This model assigns ratings to software based on the various aspects of security, performance, scalability, architecture, functionality, quality, support, community, documentation, adoption and professionalism. A weighed mean of the various ratings is the rating for the software. This can be also used to compare two open source software to decide which one of them is better suited in a certain scenario.

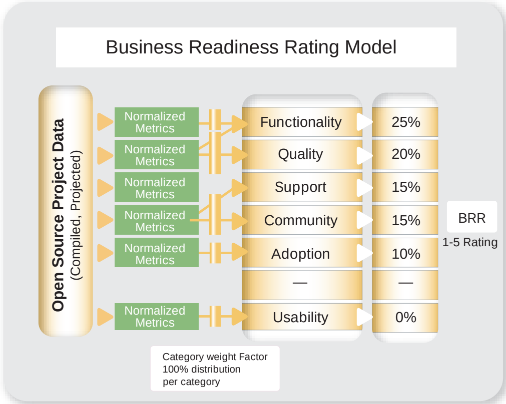

Figure 4 The BPR rating calculation

SPE and OSS - The Synergy

We at VAssure, use a lot of Open Source frameworks and tools and they add a lot of value to our applications. Besides the fact that they come free, they also add a lot in terms of extendibility, reliability and flexibility in the products. We have consistently given our partners very well received solutions by the use of these.

The OSS advantage

VAssure uses Open Source software to enhance its delivery and products and this has been our winning edge in many cases. It has sometimes meant fixing some problem in one of the modules of the code or extending functionality to suit our needs and mostly the effort has been worth the results. Any queries or bugs reported get suitably quick responses once posted on the forums.

For example, using Hibernate has saved the developers here a lot of effort in writing the data access layer. Also, it is far more configurable and light weight than using the entity beans framework. Similarly, using XMLBeans adds a lot of flexibility and a seamless integration to Java than using the java objects generated from xml-spy would give. Maven as the build tool has made deployment look a lot simpler.

Apart from core open source products, we also benefit from other projects like confluence, a knowledge repository cum document management portal and JIRA, a bug tracking tool which are minor variations of their open source counterparts adding a significant value for a minimal charge. Rajul Garg, one of the founders of VASSure says "with these solutions covering the market, I do not understand how will the propriety products offering nearly the same services sell?"

SPE processes plug OSS gaps

One problem faced with many open source projects is that they have innocuous bugs which might have slipped in due to insufficient testing, especially if the product is in a "just released" state. The maturity of the release process of SPE as envisaged at VAssure goes a long way in alleviating this gap.

Following a proper phased approach ensures that even if a part of software used is not completely bug-free, the bugs are caught at early stages and do not slip into the final production release.

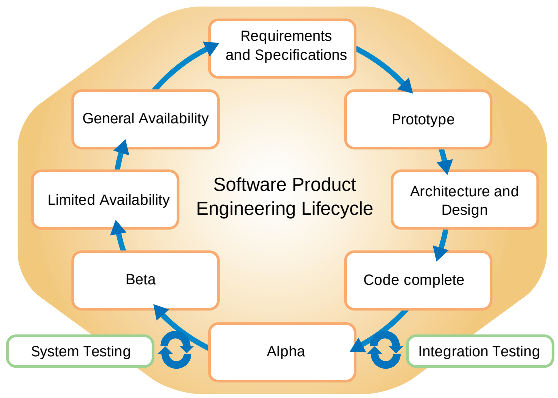

Figure5 The SPE Lifecycle followed at VAssure

Dipping Cost of Development - Solving the Right Problem

With decreasing operating margins and very informed and knowledgeable customers, it is imperative that trying to reinvent the wheel in a more efficient way is solving the wrong problem. It would rather make sense to join the reliable lot of immensely talented people and add value to the already existing solutions.

> *"There's enough room in the market for everybody."*
>
> *...Maike Balma,*
> *Linnux business strategist, Hewlett-Packard*

That would be a win-win solution for everybody and energies can be directed towards solving the actual business problems. If that means acknowledging the author or releasing the enhancement to a larger audience, that should not be a problem. In any case, this is an age when the timeliness of the product is more important and any organization which would introduce/ enhance the idea first would reap the benefits.

This is an age of creating collaborative value and the organizations which would not take this call would be left behind.

Differentiation While Using OSS

One obvious question would be that if every one is using the same OSS, what would be the differentiating points between two products. To get to the answer, we need to get into what one has to do before the OSS tool used is made business ready and also repeatable. When using Hibernate or Maven on a first project in an organization, one faces many issues and the product requires to be thoroughly understood, tweaked and documented before it becomes repeatable for use on other projects.

It requires a mindset to be able to attack the problem, find answers to questions on the forums and be able to modify and recompile binaries if required - skills hugely more rewarding but thoroughly different from those required for using propriety software. This is where SPE has the role to play and business which start using it first get the better out of it. For organizations like VAssure , after a couple of usages, it becomes a robust internal tool which can be just used off-the-shelf like any other product.

Value Added Products using OSS

An intelligent way to leverage the OSS paradigm is to create value added services like confluence on top of OSS tools and offer them to our partners as addends to the core products. Or where applicable, plug a piece of a solution applied to some other product by customizing the source code to suit the requirements of the current product. Increasingly, a pool of utilities would be created which would have taken very little time to enhance or develop (since they would be based on robust open source models) and would be completely open to customization since the one time cost of understanding the source code to modify it would already have been done.

The Path Ahead

In the time to come, companies which do not respect the fact that software can not be built for ever in the way most of it is built today; run the risk of being out performed. Businesses would have to recognize the fact that the products built should hold on to the various product engineering values. Further, in the long run, businesses which would not spend a significant portion of their time and resources to customize, enhance and use the OSS solutions run the risk of being sidelined.

That said, OSS should be used with appropriate caution and diligence. Applying the robust and very sophisticated processes of SPE to enhance and monitor OSS solutions for quality and readiness is of utmost importance. An apt synergy between the two paradigms would have an important role to play in the software industry of tomorrow.

Bibliography

1 Top Tips for Selecting Open Source Software
http://www.oss-watch.ac.uk/resources/tips.xml

2 How to Evaluate Open Source Software / Free Software (OSS/FS) Programs
David A. Wheeler
http://www.dwheeler.com/oss_fs_eval.html

3 Choosing Open Source, A Guide to Civil Society Organizations
Mark Surman and Jason Diceman Jan 06 2004
http://www.commons.ca/articles/fulltext.shtml?x=335

4 Business Readiness Rating for Open Source
www.openbrr.com

5 Massachusetts Secretary of Administration and Finance
Eric Kriss: "Open Mind on Open Source".

6 Open Source Software / Free Software (OSS/FS) References
http://www.dwheeler.com/oss_fs_refs.html

7 The Open Source Definition
http://www.opensource.org/docs/definition_plain.php

8 The VAssure Web Site
http:// www.vassure.com