Promising the Best of Both Worlds

The Impact and Future of Commercial Open Source

May 25, 2006

Evolved Media Network
242 West 30th Street, Suite 801
New York, New York 10001
dwoods@EvolvedMediaNetwork.com
646.827.2196

Introduction

The past year has seen the markets for commercial and open source enterprise software converging at an accelerating pace.

Consider:

- Established software suppliers such as IBM and Oracle each have acquired a leading open source companies–Gluecode and Sleepycat Software, respectively.
- Red Hat, perhaps the most successful open source provider so far, purchased JBoss, another open source pioneer, for $420 million.
- Venture capitalists have been investing heavily in open source companies focused on just about every category of enterprise software.

Once perceived as a nerdy phenomenon on the fringes of IT, open source is topic No. 1 in enterprise software circles, right now.

Executive Summary

- Commercial Open Source applications will have a significant and growing impact on the $60 billion enterprise software market in coming years.
- Commercial Open Source's try-before-you-buy appeal is difficult for traditional software companies to match and will affect how enterprise software is bought and sold.
- Business leaders should evaluate commercial open source solutions for mature product areas such as Customer Relationship Management (CRM), Content Management and Business Intelligence.

SugarCRM® asked Dan Woods, author of *Open Source for the Enterprise* (O'Reilly Media, 2005), to interpret this convergence and to evaluate the various flavors of hybrid commercial-open source business models that it is spawning. This paper is the result, examining for the sake of IT professionals and managers who may be unfamiliar with open source the dynamics of this historic change in how enterprise software gets designed, written, marketed, and sold.

Open Source Arrives

Until quite recently, “open source business model” was a contradiction in terms. Open source, common wisdom held, was merely the work of activists who had little interest in making a profit. Today, open source is a serious business that’s generating sizable revenues and profits while delivering significant cost advantages to customers. To understand this change is to better grasp the ways in which open source has come to be embraced by users and traditional software suppliers.

Originally, open source projects were self-organizing communities of programmers who worked together, via the Internet, on creating software that was important to them. Each community worked on the design, authoring, and on-going maintenance of a particular program. Open source software is frequently also called Free and Open Source Software to acknowledge the indispensable role of the Free Software movement in laying the foundation. Richard Stallman, founder of the Free Software Foundation, created the GNU project which provided the tools for software development on which most of the world of open source is constructed. The term “open source” was created by a set of activists who wanted to separate themselves from what some saw as the overly strident leadership of the Free Software Foundation. From these beginnings, many open source projects thrived, often involving thousands of individuals, each person contributing as much as his or her skills and schedule permitted.

Today, open source is a serious business that’s generating sizable revenues and profits while delivering significant cost advantages to customers.

This phenomenon’s first and most visible effect on the commercial software market was the widespread acceptance of the Linux operating system, which was created from the foundation provided by the GNU project. Linux was soon followed by other key blocks of systems software, together forming a key structure for corporate computing, the so-called LAMP technology stack: the Apache web server, the MySQL database, and languages such as Perl, PHP, and Python.

Commercializing Open Source

Here, large enterprise software companies and their customers recognized, was an alternative source for many important pieces of software infrastructure, software that was often technically superior to the dominant brand and always lower in cost. IBM, SAP, Oracle, and many other software companies soon found ways of using open source to their advantage.

An early adherent to this new business model was Netscape, which released its Web browser into the public domain so that the software would benefit from the contributions of as many people as possible. Today’s increasingly popular Firefox browser is the descendant of that joint effort. In the meantime, many other software companies have followed suit, releasing established products as open source code in order to keep them alive, to avoid leaving customers in the lurch, and to lower development costs.

As open source software matured, it enabled Yahoo!, Google, and Amazon to create large businesses. Enterprise software makers realized that they should start using open source projects as components in their own products. Apache quickly became the dominant web server, for instance, and companies such as SAP and Oracle certified their products for use on Linux. IBM and Dell, among others, also began to provide Linux as a pre-installed, formally supported option on PCs and servers.

Several Generations of Hybrid Models

One of the earliest opportunities in commercializing open source projects was identified in the area of technical support. Companies like Red Hat and Suse, now a part of Novell, set out to improve upon the limited support that the open source community was providing for Linux and thereby make the software more attractive to enterprise customers. Their approach was to sell subscriptions to their own versions of Linux, each of which, for a price, provided added functionality such as installation scripts, collections of pre-tested open source add-ons, streams of updates, and full maintenance and support.

Next came companies launched specifically with the intention of acting as commercial software firms but making open source licensing and practices central to their strategies in product development, support, distribution, and marketing. JBoss, starting out to make an application server and later creating an enterprise-class Java development environment, is a prime example, as its direct competitor Gluecode. In 2005, IBM purchased Gluecode and in early 2006, Redhat disclosed plans to buy JBoss.

Expanding Hybrid Models to Applications

Until recently, almost all hybrid open source projects focused on infrastructure and development tools. But now, enterprise applications software, the kind that directly addresses business problems, has become the focus of commercial open source, too.

Enterprises have long enjoyed the comfort and safety of licensing traditional enterprise software. They've come to depend on these products to automate vast swaths of their activities, from financial accounting to managing supply chains. And software companies now supply specialized programs for just about every vertical market under the sun.

Yet, there are significant complaints from parts of the market. Not surprisingly, customers would like to pay less, and enterprise software suppliers have responded in many different ways seeking to reduce the total cost of ownership of their products. Some customers are wary of getting too dependent, or locked in, to a particular software system.

The past few years, however, have companies such as SugarCRM, JasperSoft, Alfresco, Pentaho and Zimbra use the open source model to address these and other less-than-pleasing aspects of traditional enterprise software products. In general, these companies take the following approach:

- A simplified but functionally complete version of the product is available as open source.
- For a fee, a premium version of the product with advanced features is available, usually with access to the source code.
- Documentation may be free or may be available for a fee.
- Community support is available for free, as with other open source projects.
- Community members offer enhancements to the open source version of the product.
- For a fee, access to enterprise quality support services such as phone support is available.
- Development and implementation consulting is available for a fee from the company directly or from partners in the ecosystem.

Commercial open source applications are the most complete blending, yet, of traditional software business models and open source practices.

It expands the scope of the hybrid model beyond infrastructure-level software.

Commercial open source applications are the most complete blending, yet, of traditional software business models and open source practices. It expands the scope of the hybrid model beyond infrastructure-level software. But this new model comes with its own challenges.

A key question has been raised by potential customers, industry analysts, and venture capitalists looking at open source as an investment theme: Why, they ask, has it taken so long for the application hybrid model to emerge?

In fact, awareness of this new model is still emerging and it is likely that until now, entrepreneurs have thought that raising and earning money would be easier with the traditional enterprise software business model. But the idea that a hybrid software business can be built by giving a large portion of a product away as open source and then supporting it like commercial software is taking hold. The rest of this paper will examine when and where hybrid models are likely to thrive.

Benefits of Hybrid Models

The fact is, hybrid models yield benefits for both users and suppliers of software. For customers, the most obvious and attractive benefit is lower prices. Hybrid software costs them less to acquire than commercial alternatives. One important reason: Hybrid companies have lower costs than their traditional counterparts. They usually accelerate their development efforts by using open source components and by initially mimicking the requirements of mature products. They also rely on viral marketing instead of more-expensive outbound sales efforts. Their primary marketing vehicle is the quality and easy availability of their basic, open source product.

Indeed, a big benefit for users is that they are able to download and try out the open source version of even a highly sophisticated software product at essentially no charge. And if this software meets their needs and they can support it themselves, perhaps with help from the project community, users may never have to pay anyone a dime. Yet, even if they ultimately intend to pay for a fully supported product, they can still install the basic software, configure it, and let end-users try it out at hardly any cost.

This try-before-you-buy approach encouraged reduces risk in a few additional ways. The customer gets a hands-on feel for the performance of the software, for its configurability, and for how well it matches the skill levels of his or her IT staff. If those are sufficient, then there's no need to pay for formal support. But where the IT staff lacks skills or time, then support is available for a fee, that is frequently lower than the support charges for enterprise products.

When a company wishes to extend or modify the software in a deeper, more extensive manner, open source gives them the ultimate freedom.

The configuring of hybrid products relies on the same techniques used in commercial enterprise software: APIs, templates, XML files, and so forth. But when a company wishes to extend or modify the software in a deeper, more extensive manner, open source gives them the ultimate freedom: Its source code is available for unrestricted modification. While customers take advantage of this capability only infrequently because of the development skill required, just knowing that it's possible can comforting. This is, perhaps open source's response to that much-touted benefit of traditional software; that it offers "one throat to choke."

There's comfort, too, in the community that grows up around hybrid open source projects. Typically, these communities are quite similar to those driving traditional open source projects: Enthusiastic users share their ideas and actual code extensions with the community, creating new features and translating user-interfaces and documentation into new languages, for instance. The community also provides a mechanism for influencing the future direction of the product. Users are able to find each other in an unrestricted fashion so as to share their experiences with using, implementing, and maintaining the software. Because users and developers of the hybrid are able to look at the same code base and inspect each other's efforts, communication between the two groups is often quite direct.

The benefits for vendors? They pretty much mirror those of the customer. The project community helps with the gathering of requirements and the testing of new additions to the software. Users' freedom to download and use the open source version in actual business settings greatly accelerates the sales cycle and reduces marketing costs. As a result, the marketing budgets at hybrid companies tend to be much smaller than those of traditional software firms, thereby freeing more resources for product development.

Current and Future Impact of Hybrids

It is fair to speculate that hybrid business models will have a significant and growing impact on the $60 billion enterprise software market in coming years. Their inherent economic benefits will give companies that adopt these hybrid models a price advantage that will be used to attack the market share of the larger companies, which have size and longevity working in their favor.

One of the key breakthroughs that hybrid applications models will achieve is to assuage the understandable fear and uncertainty that many companies, especially small and medium-sized businesses, have about open source. If open source is a pathway to skill-building, as the book *Open Source for the Enterprise* argues, what happens if the skills develop slowly or not at all? With traditional open source projects, the failure to create skills can cause an entire project to fail. With a hybrid model, in contrast, a thriving, for-profit organization can take up the slack and provide any level of support that may be required. Indeed, this arrangement alone will ease the way for ranks of smaller businesses, which are financially unable to maintain high-skill IT operations, finally to avail themselves of open source's many technical and economic advantages.

With traditional open source projects, the failure to create skills can cause an entire project to fail. With a hybrid model, in contrast, a thriving, for-profit organization can take up the slack and provide any level of support that may be required.

Hybrid business models will certainly affect the selling of enterprise software, too. Open source's try-before-you-buy appeal is difficult for traditional software companies to match. If hybrid market share continues to grow, it will not be surprising to see them start to offer trial versions of their products as a way to reduce marketing costs and speed up notoriously long and costly sales cycles.

From a broader market perspective, the key question is, How much of the software business can the hybrid model capture? The large enterprise software firms emphasize their ability to offer integrated suites of software for ERP, CRM, SCM, and so on. Most hybrids offer a single product, or occasionally two integrated applications. If customers insist on suites, then hybrids will face a greater challenge. Hybrid companies argue that many customers will reject the suite approach in favor of choosing the best product for a particular purpose, especially when faced with the lower costs for hybrids. In the end, the resolution of the suite vs. best of breed question will depend on whether the benefits of the integration of the suites overcome the cost of integrating hybrids with the rest of a company's systems.

It is likely that the hybrid model will find its greatest success in mature categories of software. There, requirements are well understood, and the challenge for suppliers boils down to replicating a set of common requirements, not inventing new ones.

How would a hybrid model succeed in an area that has never before been automated application area. The hybrid company eyeing such a market would have to create software that met a new need, and then give it away. If the need for the software is so compelling as to warrant a new product, why not just sell it to begin with? The reason that hybrid software is given away in current markets is that it is competing against more expensive alternatives. In the absence of those expensive alternatives, does the hybrid model make sense?

The other question looming over the hybrid model concerns how a company's founders are to be rewarded. If a team of experts has a great idea or great skills to create an enterprise application, why use the hybrid model with its reduced expectations for revenue unless the reason is to attack an existing market with a lower cost product.

Taken together, the characteristics of hybrid models suggest that they will be used primarily to attack mature, established product areas with lower-cost products. In this way, hybrids will accelerate the commoditization of these product areas. New areas of innovation in enterprise software are not likely to be pursued as aggressively with the hybrid model.

Perhaps there will emerge some sort of indicator of a product area's maturity based on what portion of the premium product is available as open source. In brand new areas, perhaps only a small fraction of the product will be available in that form. In maturer areas, that percentage will be driven higher as hybrid software suppliers enter the market.

One key indicator of the impact of hybrids will be the number of companies that get funded to follow in the footsteps of firms like SugarCRM. The more such companies emerge and succeed, the more dramatic will be open source's impact on the 35-year-old enterprise software market.

About The Author

Dan Woods, CTO and Publisher, Evolved Media Network.

Dan Woods has a background in technology and journalism. He has a BA in Computer Science from the University of Michigan. He was CTO of TheStreet.com, CapitalThinking, led development at Time Inc.Ãs Pathfinder, and created applications for NandO.net, one of the first newspaper web sites.

Dan has a MS from Columbia UniversityÃs Graduate School of Journalism. He covered banking for 3 years at The Record of Hackensack, was Database Editor for three years at the Raleigh News & Observer, and has written two books on technology topics, in addition to numerous white papers and magazine articles.

About SugarCRM

SugarCRM is the world's leading provider of commercial open source customer relationship management (CRM) software for companies of all sizes. SugarCRM's mission is to provide the most innovative, highest quality, easily customized and most appreciated business tools created by the open source development and business models.

SugarCRM develops solutions by actively engaging the CRM community, consisting of users, customers, developers and experts, and incorporates their needs, opinions and experiences into the solution. SugarCRM conveys our strengths, weaknesses, vision and function in a forthright manner to all community members so that the solution expectations are not misguided. SugarCRM believes that our open source development methodology and open source business models maintain healthy business relationships, highest quality solutions, and also the highest level of trust.