

May 20, 2004

Applying Open Source Processes In Corporate Development Organizations

by Liz Barnett

BEST PRACTICES

BEST PRACTICES



May 20, 2004

Applying Open Source Processes In Corporate Development Organizations

By **Liz Barnett**

With Carey E. Schwaber

EXECUTIVE SUMMARY

Open source projects have resulted in many of today's most innovative new products: operating systems, application servers, Web servers, and databases. What is it about the open source development model that works so well? Why don't corporate IT shops have the same types of successes? Many of the practices and staffing models used by open source projects are relevant to corporate IT; managers and developers should study and adopt these. Other open source strategies are not unique — in fact, many are also principles of Agile development and have been proven to scale on large distributed open source projects. These, too, are largely applicable to a corporate development team and should be incorporated into development processes. However, there are a few open source techniques that would be difficult, if not impossible, to implement in a corporate environment; rather than consider these, managers should work to improve their existing techniques and accept the constraints that corporate culture can impose.

TABLE OF CONTENTS

- 2 **Corporate Development Processes Have Room For Improvement**
 - 2 **Adopt These Open Source Development Processes**
 - 6 **Adopt Agile Processes Being Proven Out On Open Source Projects**
 - 8 **Avoid These Open Source Processes**
- RECOMMENDATIONS
- 9 **Corporate IT: Leverage Open Source Development Techniques**
 - 11 **Supplemental Material**

NOTES & RESOURCES

Forrester interviewed vendor and user companies, including: Capital One, Groove Networks, Hewlett-Packard, Olliance Group, PlanetOut, The Open Group, and Three Rivers Institute. Forrester also spoke to researchers at Harvard Business School.

Related Research Documents

"Adopting Agile Development Processes"
March 25, 2004, Trends

"Your Open Source Strategy"
September 23, 2003, Report

"Executive Overview: Linux And Open Source"
January 24, 2003, Brief

CORPORATE DEVELOPMENT PROCESSES HAVE ROOM FOR IMPROVEMENT

Today's corporate software development is broken: Less than a third of projects finish on time, on budget, and within scope, and the overall percentage of functionality delivered is barely more than half.¹ What's being done about this? Top-down process improvement initiatives have been implemented in many IT organizations, but they haven't won developers' hearts and minds. Agile development processes have gained a considerable following — especially among developers — but skeptics still doubt their scalability. The open source development model, in contrast, has been proven to work on large-scale projects and has enticed hundreds of thousands of volunteers to work on open source projects for free (see Figure 1).²

Corporate Development Can Learn From Open Source Development

Open source software has moved from the margins to the mainstream; 46% of firms Forrester surveyed currently use open source and another 14% plan to use open source within the next 12 months.³ Many open source projects have achieved success — Apache, Linux, MySQL, and Tomcat, for example — winning worldwide renown for their open source development models. Corporate IT should respond by:

- **Adopting some open source processes.** Many open source processes are suitable for corporate environments and can help improve developers' productivity and quality.
- **Adopting some Agile processes proven out by the open source community.** Open source projects have demonstrated how a number of Agile techniques, such as formal user involvement and continuous integration, work for large-scale efforts.
- **Avoiding other open source processes.** Some aspects are particular to open source and simply don't translate in a corporate or government development environment.

ADOPT THESE OPEN SOURCE DEVELOPMENT PROCESSES

There are at least three areas in which open source practices can benefit corporate IT organizations:

- **Team communication.** IT teams should emulate open source developers' frequent online communication. Open source teams recognize the need to communicate across time zones, cultures, and languages.
- **User involvement.** In open source development, users are inherently involved in development, since the developers typically *are* the users.

Figure 1 Comparing Traditional, Agile, And Open Source Development

	Traditional	Agile	Open source
Documentation	Documentation is emphasized as a means of quality control and as a management tool.	Documentation is deemphasized.	All development artifacts are globally available, including code and information documentation.
Requirements	Business analysts translate users' needs into software requirements.	Users are part of the team.	The developers typically are the users.
Staffing model	Developers are assigned to a single project.	Developers are assigned to a single project.	Developers typically work on multiple projects at different levels of involvement.
Peer review	Peer review is widely accepted but rarely practiced.	Pair programming institutionalizes some peer review.	Peer review is a necessity and is practiced almost universally.
Release schedules	Large number of requirements bundled into fewer, infrequent releases.	Release early, release often.	Hierarchy of release types: "nightly," "development," and "stable."
Management	Teams are managed from above.	Teams are self-organized.	Individual contributors set their own paths.
Testing	Testing is handled by QA staff, following development activities.	Testing is part of development.	Testing and QA can be performed by all developers.
Distribution of work	Different parts of the codebase are assigned to different people.	Anyone can modify any part of the codebase.	Anyone can modify any part of the codebase, but only committers can make changes official.

Source: Forrester Research, Inc.

- **Staffing models.** Open source developers work on multiple projects at once, but are more involved with some than with others. IT managers should strive to support this model.

Build A Robust Team Communications Infrastructure

Open source projects rely heavily on tools as modes of communication (see Figure 2-1). For instance, open source projects are more likely to build and use automated communications archives. Mailing list archives are the most commonly used method, although wikis — Web-based collaborative writing tools — are growing in popularity. Corporate IT shops must strive to capture and store various team communications — from emails to IMs to virtual conferences — to improve team coordination (see Figure 2-2). Scalable collaborative development environment like CollabNet SourceCast and VA Software SourceForge can help. This approach will:

Figure 2 Development Team Communications Tools

2-1 Open source team communications tools	
Collaborative environment	SourceForge
Technical discussion	Wikis, mailing lists, Web sites, FAQs
Version control	CVS and Subversion
Issue tracking	Bugzilla, Scarab
Build systems	Anthill, CruiseControl, Gump, Make, Maven, Tinderbox

2-2 Team communications infrastructures in practice at corporate IT shops	
Groove Networks	Groove uses blogs to create shared knowledge bases and keep employees apprised of the company's progress toward goals.
PlanetOut	The development organization logs its communications in a wiki.
Hewlett-Packard	HP's use of CollabNet is so sophisticated that its developers essentially work virtually, automatically documenting their communications on the network.

Source: Forrester Research, Inc.

- **Make documentation less of a chore.** These mechanisms automatically document parts of the development process, freeing team members for more valuable tasks. While team communications systems won't do away with documentation altogether — development teams will still have to create artifacts like requirements documents, design models, and test plans — these tools will cut down on the amount of documentation the team must explicitly create.
- **Create a valuable information asset.** Archiving as much of the development process as possible also protects a team's knowledge base, which is susceptible to depletion in the event of employee departures. Even sick days are less problematic at firms that use this strategy. Automatically logging development activities on the system is an excellent way to capture and transfer knowledge among team members.
- **Increase transparency.** Firms should increase transparency — the visibility of the project information to people who aren't team members — to improve coordination and help increase productivity. Increased transparency can also improve the accountability and the performance of individual employees by providing a documented workflow and a basis for feedback. Leading development organizations know that disclosure improves business relationships and will communicate select information about project progress to customers.⁴

Involve Users In Development

Linus Torvalds observed that a large user base can be even more valuable than a large developer base. Eric S. Raymond posits that “given enough eyeballs, all bugs are shallow.”⁵ Not only do open source users specify requirements but they also participate in design reviews, testing, and implementation of new systems. IT organizations should look for ways to involve end users in as many stages of development as possible. For example, firms should:

- **Reveal development artifacts to internal users.** Make source code and other artifacts, such as requirements documents, issues lists, and project schedules, available to internal customers — especially those with technical savvy — whenever possible. This will allow customers to identify issues earlier in the life cycle and to feel a greater sense of ownership of the overall project.
- **Recruit a crew of beta testers.** Select from among your most involved customers a few who are willing and able to use beta versions of your application. This can work for development organizations of all levels; IBM and Microsoft involve beta testers through the Microsoft Developer Network (MSDN) and IBM’s alphaWorks. Corporate IT shops should follow suit on a smaller scale.⁶

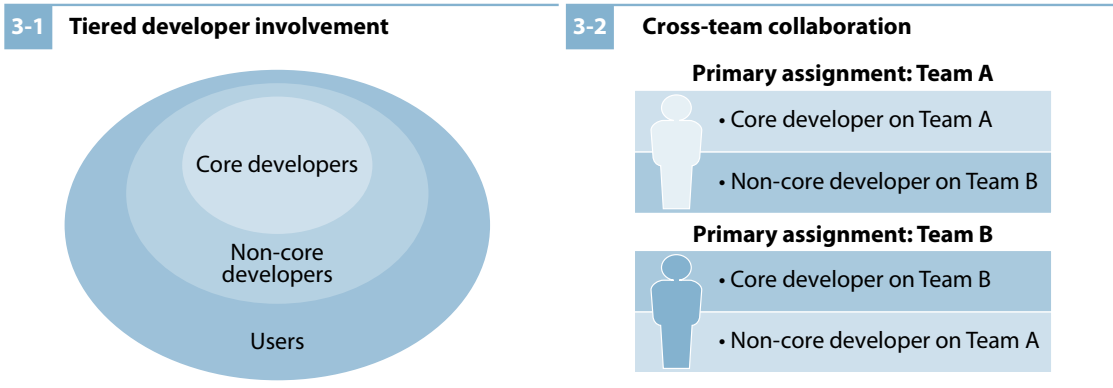
Staff Teams Flexibly And Cross-Functionally

Open source projects typically consist of a small number of core developers, a much larger number of non-core developers (contributors), and a vast pool of users (see Figure 3-1).⁷ More than half of open source developers participate in two or more projects, and another 10% participate in 10 or more.⁸ Developers benefit from the broader exposure that results from their involvement on multiple projects; they are generally more engaged and more satisfied than their peers.⁹ IT organizations should organize themselves to achieve these benefits by:

- **Injecting an element of developer self-determination.** Ten years ago, a Fortune 100 company’s IT shop launched an innovative staffing process: Mentors worked with developers to determine where developers’ talents and interests lay and then negotiated to staff them on the most appropriate projects.

The upshot was that the retention rate grew at a time when client/server developers were in great demand and were being wooed by many other firms. Developers preferred this bottom-up staffing process because it gave them some control over their fate. Managers ended up leading teams of people who were more likely to be excited about their work.¹⁰

Figure 3 Experiment With Tiered Levels Of Developer Involvement



Source: Forrester Research, Inc.

- **Forming teams of core and non-core developers.** Communicate to programmers that they are the core developers on their primary teams but that they should function as non-core developers — that is, contributors — on other teams (see Figure 3-2).¹¹ Cross-staffing exposes developers to new methods. Plus, when developers are on multiple teams, it's more likely that they'll be asked to do the work that they can do best instead of whatever work is available.
- **Upgrading the role of code committer.** In open source communities, the most dedicated and accomplished members of a project are entrusted with the job of committing new or enhanced code to the code repository. Corporate IT shops should transform this from an administrative role to one of coveted leadership by endowing it with additional responsibilities and clout, similar to those of the open source committer. The code committer can also be responsible for enforcing peer code review and for maintaining the modularity of the system — even when the pressure's on.
- **Revising developers' metrics accordingly.** Companies that allow developers to choose which other teams' projects to work on must revise developer goals to reflect this involvement. Managers must incent developers based on how much code they contribute to other projects and how much of that code is accepted. The success of this strategy requires that developers clearly understand that they are expected — not just encouraged — to contribute to more than one project. All levels of management must be onboard for this strategy to succeed.

ADOPT AGILE PROCESSES BEING PROVEN OUT ON OPEN SOURCE PROJECTS

Many of the principles of open source development match those of Agile development processes, such as eXtreme Programming (XP) and Scrum.¹² And while adoption of Agile

development processes is increasing, there are still many who doubt that they are practical for large projects. In fact, many open source projects have demonstrated that Agile techniques can work on a large-scale, distributed basis. Corporate IT teams should evaluate the following practices that are common to open source and Agile development:

- **Release software early and often.** Open source and Agile development both use shorter, more frequent release schedules.¹³ This strategy works because it delivers value to customers early on, rather than making customers wait until the full release before beginning to capture value. Another benefit of this practice is that customers that are using early versions of applications are able to provide more timely feedback. At Hewlett-Packard, manufacturing clients inspect software as early in the cycle as possible and provide feedback on how the software will — or won't — work in their organizations.
- **Formally involve users in development.** Users' deep involvement in development is often cited as one of the primary strengths of the open source model. Traditional user involvement occurs mainly in the early stages of a project (e.g., requirements gathering) and the final stages (e.g., user acceptance testing). Agile processes require that users be involved throughout the project, and open source developers continually represent users' interests. To make this happen, IT teams must work with the business community to impress upon them the value of their ongoing involvement. A number of Agile teams have also cited their users' improved perception of the IT teams — they feel more involved in the process and more in control of the final deliverable.
- **Enact collective code ownership.** Collective code ownership — a cornerstone principle of both open source development and XP — allows any developer to change any part of the system, whether or not he or she has explicitly been tasked to do so. Allowing developers to improve their peers' code when they see opportunities to do so — and when they have the appropriate subject matter expertise — improves the overall quality of the codebase. Furthermore, it prevents any one developer from becoming a bottleneck. With collective code ownership, the entire team is responsible for the entire system, affording developers responsibility as well as freedom.
- **Practice continuous integration and employ automated testing environments.** Continuous integration and automated testing, two software engineering practices that XP stresses, are de rigueur in open source development — in part because it is more challenging to integrate the work of open source developers than that of corporate developers. Open source uses build tools like Anthill, CruiseControl, Gump, Maven, and Tinderbox. The best among these take code out of a repository, compile it, rerun a series of test scripts, generate bug lists, and feed the results back into the bug tracking tools, community mailing list, and Web site. Commercial alternatives from firms like

Mercury Interactive, Compuware, and IBM/Rational do a good job at automating regression and load testing, but they aren't as tightly integrated into the software configuration management's build management capabilities.

Open Source Follows Some Rules Better Than Corporate IT Does

A number of classic software engineering practices are more commonly used by open source developers than by corporate developers. These practices solve problems that are more threatening to open source projects than to corporate IT projects. Open source projects are consequently more likely to:

- **Enforce peer code review.** Peer code review is a fundamental principle — and strength — of the open source development model.¹⁴ Widely acknowledged to result in better code, peer review nonetheless might fall by the wayside come crunch time. Why does it work in the open source model? Two reasons: 1) When the identity of contributors is largely unknown, it's more important to review code before committing it, and 2) open source code is more likely to be modular, which means that the code committers at the intersection of each branch have the ability to enforce code review.
- **Construct a minimal code base and add new functionality in separate modules.** Few would dispute that modular software development is better software development; modularity has been a best practice since the days of structured design methods. Best practice or not, modularity doesn't get the attention from corporate IT that it deserves. Modularity works in the open source world because developers know that they will have to work independently of each other without jeopardizing the stability of the system. They must build functionality in small increments, adding new modules as the open source project matures.¹⁵ The distributed, decentralized, and asynchronous nature of open source development means that teams must rely on clear and clean modularity to ensure that separately developed features and functions hang together. In short, open source development forces the issue on modularity.¹⁶

AVOID THESE OPEN SOURCE PROCESSES

Unfortunately, some of the processes that make open source development so successful are difficult — if not impossible — to port over to corporate development environments. While it's easy to change tools or even some development practices, it's far more difficult to change management philosophies. For example, it's unlikely that a corporate development would allow the following open source practices:

- **Developers chart their own courses.** Open source developers cite the ability to work on the projects that interest them as one of the best things about the open source

model. Unfortunately, most IT shops are so pressed for staff and budget that flexible staffing isn't feasible for them. If at all possible, though, firms should allow developers to choose even a small portion their work. 3M famously required its engineers to spend 10% of their time on projects of their own choosing. One Fortune 100 firm's development organization formerly dedicated 20% of developers' time to personal development — that is, non-project-specific activities. And a prominent middleware tool vendor considered allowing its developers to spend a fraction of their day dabbling in the projects that interested them most.

- **Developers are driven by intrinsic — not extrinsic — motivations.** It's the rare corporate development organization that can inspire the kind of dedication found among open source developers. Members of open source communities are driven to a large extent by the desire to gain recognition. This type of visibility isn't possible in an IT shop because the end user community is smaller and is unlikely to appreciate high-quality code. Corporate developers are more commonly motivated than open source developers by career goals and related benefits.

As open source successes become more well-known and result in products that are used in corporate shops, managers will find their staff to be more engaged with open source practices. Managers who aren't open to new techniques risk either losing developers to more innovative organizations or retaining developers with poor morale. Managers who allow developers to bring their open source experiences to bear in the corporate environment stand to increase both software quality and employee satisfaction.

RECOMMENDATIONS

CORPORATE IT: LEVERAGE OPEN SOURCE DEVELOPMENT TECHNIQUES

- **Automate parts of the documentation process.** Development organizations, especially those working in distributed or decentralized environments, require high levels of communication and collaboration. Teams should investigate the open source tools available to them, such as wikis and Bugzilla. If these tools don't cut the snuff, though, firms should investigate more comprehensive offerings like SourceCast and SourceForge.
- **Get users as involved in development as possible.** Users have historically hated being involved in development projects, in large part because it takes them away from their day jobs. But Agile and open source development provide users with tangible and frequent rewards for their involvement. Corporate IT teams should ensure that users receive concrete benefits from their involvement. Incremental iterative development makes this possible.

- **Tier developer involvement on multiple projects.** Whenever possible, allow developers to participate in more than one project to increase collaboration and, therefore, innovation. Assign developers different levels of involvement and responsibility on their various projects.
- **Explore Agile techniques that have been proven to scale effectively.** Adopt the planning and engineering practices that open source has in common with Agile development: Release early and often, have a customer representative on the project, and implement test-driven development. Adopting Agile development processes is the simplest way to make your organization more like an open source community — and to reap the accompanying rewards.

SUPPLEMENTAL MATERIAL

Online Resource

A repository of relevant articles is available at MIT Sloan's Free/Open Source Research Community (http://opensource.mit.edu/online_papers.php).

Companies Interviewed For This Document

Forrester interviewed software development thought leaders, including Carliss Y. Baldwin of Harvard Business School, Kent Beck of Three Rivers Institute, Kevin Crowston of Syracuse University, and Ken Schwaber of Advanced Development Methods. Forrester also interviewed decision-makers at the following companies, as well as companies that preferred not to be named:

Capital One	Olliance Group
Groove Networks	PlanetOut
Hewlett-Packard	The Open Group

ENDNOTES

- ¹ Source: CHAOS Chronicles v3.0, The Standish Group, 2003, www.thestandishgroup.com. The Standish Group's CHAOS report presents findings on project success and failure rates.
- ² There is no single open source development model. Each open source process has its own development process. For the purposes of this research, we have examined processes common to most or all open source projects.
- ³ Forrester fielded an online survey to 140 US and Canadian companies that belong to the Forrester Executive Research Panel. We heard from a mix of companies: Fifty percent with revenues more than \$1 billion; 20% with revenues between \$500 million and \$1 billion; and 30% with revenues less than \$500 million. See the March 16, 2004, Trends "Open Source Moves Into The Mainstream."
- ⁴ Preemptive disclosure is one of the cornerstones of Forrester's Organic Business strategy. For example, a smart Hungarian hard drive maker gives IBM direct access to the quality metrics of the drives coming off its line, forcing the manufacturer to fix problems before they reach the loading dock. The moral of the story? If your customers are going to find out eventually, it's better to share information preemptively so they become addicted to your candor. See the March 18, 2004, Forrester Big Idea "Organic Business."
- ⁵ Source: Eric S. Raymond, "The Cathedral and the Bazaar," O'Reilly & Associates, 2001. Selections available at <http://www.catb.org/~esr/writings/cathedral-bazaar/>.
- ⁶ One of the key elements that will make developers feel that they are part of a developer program is insider access. A popular form of insider access is availability of prerelease, or even beta,

code. IBM, in particular, provides very early access to code that is sometimes barely out of the labs through its alphaWorks site. Giving developers early access to code plays to their intense curiosity, but it can also better position them to advise their managers on upgrade decisions and timing — and anything that makes developers look good is surely in the vendor's interest. See the February 25, 2004, Best Practices “How To Win Developers And Influence Buyers.”

- ⁷ A study of the Apache project found that development activity was stratified: Core developers perform about three-quarters of development work, non-core developers perform the remainder of the development, and members of the vast user base report problems, often accompanying their reports with suggested fixes. See Audris Mockus, Roy T. Fielding, James Herbsleb, “A Case Study of Open Source Software Development: The Apache Server,” June 2000. See <http://opensource.mit.edu/papers/mockusapache.pdf>.
- ⁸ Source: Walt Scacchi, “Socio-Technical Interaction Networks in Free/Open Source Software Development Processes,” April 2004, <http://www.ics.uci.edu/%7Ewscacchi/Papers/New/STIN-chapter.pdf>.
- ⁹ Source: Telephone conversation with Rachele Rowland and Stormy Peters, April 28, 2004. Hewlett-Packard aims to assign its software engineers to multiple projects for these reasons.
- ¹⁰ The firm's program ended after its executive sponsor departed for another organization. Management support is crucial to the success of programs such as these. Not long after the program was discontinued, the development organization's retention levels decreased.
- ¹¹ “One thing we find with respect to participation is that in a couple of other surveys, 60% of open source software developers who show up as core contributors tend to be contributors to two to 10 other projects.” Walt Scacchi, “Open Source Under the Microscope,” January 5, 2004, CNET News.com. See http://news.com.com/2008-7344_3-5133553.html?tag=guts_bi_7344.
- ¹² Under the Agile umbrella lie a number of different techniques for development, modeling, and project management; it's important to distinguish between them when considering alternatives. Both corporate IT and ISV development teams are finding that Agile processes deliver business value more quickly, help improve overall efficiency and quality, and enhance team morale. AD organizations should consider Agile processes to be complementary to their existing methodologies and processes and should look for opportunities to leverage Agile techniques. See the March 25, 2004, Trends “Adopting Agile Development Processes.”
- ¹³ On average, Agile development processes suggest a 30-day release cycle. This can be a sizable challenge to an IT shop that typically delivers software on 12 or even 18 month cycles. See the March 22, 2004, Trends “Determining Iteration Length In Agile Development.”
- ¹⁴ Open source software is built using the scientific method — including published results and peer review. Doing so avoids falling into the commercial software trap of shipping products with known bugs to gain time-to-market advantage. See the September 23, 2003, Report “Your Open Source Strategy.”

¹⁵ Modularity enables parallel work; the compatibility of this work is ensured by instituting and following design rules. See Carliss Y. Baldwin and Kim B. Clark, “The Option Value of Modularity in Design,” May 16, 2002. See <http://www.people.hbs.edu/cbaldwin/DR2/DR1Option.pdf>.

¹⁶ “A small team is unlikely to enforce design rules. It’s too easy for a group of six to eight people in close contact with each other to deviate from modularity. It does seem to be a consistent theme in software development that in such cases modularity has to be enforced. It can be enforced by a manager like Fred Brooks at IBM, but it also helps if physics does some enforcing as well: If developers aren’t not in contact with each other, they aren’t able to confer, so they have to rely on design rules.” Carliss Y. Baldwin, Harvard Business School, April 13, 2004.

FORRESTER

Helping Business Thrive On Technology Change

Headquarters

Forrester Research, Inc.
400 Technology Square
Cambridge, MA 02139 USA
Tel: +1 617/613-6000
Fax: +1 617/613-5000
Email: forrester@forrester.com
Nasdaq symbol: FORR
www.forrester.com

Research and Sales Offices

Australia	Japan
Austria	Korea
Brazil	The Netherlands
Canada	Poland
France	United Kingdom
Germany	United States
Hong Kong	Spain
India	Sweden
Israel	

*For a complete list of worldwide locations
visit www.forrester.com/about.*