Proceedings of the 24th International Conference
on Software Engineering (ICSE'02)

Accelerating Software Development Through Collaboration

Larry Augustin, Ph.D.
CEO
VA Software Corp.

Dan Bressler
Director, Product Marketing
VA Software Corp.

Guy Smith
Product Consultant
Silicon Strategies

1. ABSTRACT

In early 1999, VA Software launched a project to understand how the Internet development community had been able to produce software such as Linux, Apache and Samba that was generally developed faster and with higher quality than comparable commercially available alternatives [1,2,3,20]. Our goal was simple: determine how to make more software development projects successful.

We discovered that successful Internet community projects employed a number of practices that were not well characterized by traditional software engineering methodologies. We now refer to those practices as Collaborative Software Development or CSD. Late in 1999 we developed the SourceForge platform to make it easy for even small software development projects to employ those practices, and in November of 1999 launched the SourceForge.net web site based on the SourceForge platform.

The site was an overwhelming success, and in less than two years, grew to support more than 27,000 software development projects and over a quarter million software developers worldwide. SourceForge.net affords us an unequaled test bed for understanding CSD. In response to demand from companies seeking to enable CSD within their organizations, we announced a commercial version of the SourceForge platform, SourceForge Enterprise Edition, in August 2001.

This paper describes the principles of CSD, the software development pain points those principles address, and our experience enabling CSD with the SourceForge platform.

2. INTRODUCTION

The success of Open Source software has baffled many industry observers. Open Source rapidly changed the paradigm of software development and markets by delivering more software with better quality in less time – a feat most software driven organization would like to achieve.

In 1/3 the time that it took UNIX to meet enterprise needs, Open Source has provided a scaleable and stable operating system (Linux), viable and intuitive GUIs with competent desktop metaphors (KDE and GNOME), commercial grade database systems (PostgreSQL, MySQL), and an array of solutions from office suites to enterprise applications. All this was accomplished without traditional budgeting, staffing or unified vision statements.

At VA Software, this process intrigued us. Given our origins as a premiere Linux expertise company, Open Source development disciplines were important to us. The most interesting aspect of Open Source was that their development disciplines, which seemed so removed from traditional software development practices, were in some ways more successful. We wanted to understand how far flung, ad-hoc groups of developers could successfully establish projects, execute them, and in the process produce more products more quickly and with better quality than some commercial endeavors. This ultimately lead to examining how to apply Open Source best practices to software engineering in large enterprises.

Our investigations defined Collaborative Software Development (CSD) as a concept. CSD knowledge in turn produced SourceForge.net, a CSD portal that today supports over 30,000 Open Source projects and 300,000 developers working on a vast array of Open Source projects. CSD's basic tenets were amplified within SourceForge.net, further accelerating Open Source projects.

In this paper we will discuss the principles of CSD not only as they occurred in the Open Source community, but also how they apply to enterprise software development. The same "pain points" that Open Source developers face, still exist and are growing within most organizations.

3. WHAT LIMITATIONS WERE OVERCOME

The essence of Open Source development is the rapid creation of solutions. The key word therein is "rapid". Everything about Open Source is performed in Internet time. In the Open Source community:

- Projects are established instantly
- Requirements are defined quickly
- Detail design is created and reviewed online
- Code is written by individuals or small, agile teams
- Code reviews are almost mandatory and facilitated online
- Code testing is a fast, collaborative process
- Feedback is loud and instantaneous

The common thread binding each of these elements is that the work is done in a collaborative fashion. From concept to release, two or more people work closely together through all phases of the process.[19] Without the collaborative functions facilitated by a common set of Internet technologies, Open Source would be extremely difficult if not impossible.

Wide adoption of Internet technology standards facilitated CSD and Open Source development. To accomplish feats such as Linux and KDE, the Open Source community had to overcome many obstacles that affect most enterprises as well. These obstacles include:

Recruitment: Finding the right experts to work on particular projects.

Geography: Getting people to work together from anywhere on the planet.

Communications: Facilitating both ad hoc and structured communications during each project phase.

Asynchronous: The ability to effectively collaborate across different time zones.

Common tools: Having centralized sets of tools for the management of communications, code, documentation, and knowledge.

Process: Utilizing different and flexible approaches for projects based on suitability and applicability for the project, and for the culture-fit of the team.

Project management: Process agnostic systems for managing projects and tracking the status of team efforts.

Knowledge management: Capturing and making available all the intelligence that went into a software project.

Internet technologies overcame each of these obstacles. From email to network aware code repositories, Internet tools removed barriers hindering the development process and freed developers to create superior code more rapidly.

A by-product of having highly communicative, distributed development teams was an unintentional leaning toward "agile development methods" and self-adaptive processes [4,6]. Almost universally, Open Source projects grow through small, incremental changes defined and executed by compact and highly communicative development teams. In the Open Source community, electronic collaboration replaces the need for the direct interaction required by many agile methods, most notably those proposed by eXtreme Programming (XP) advocates.

Another facet of agile development found in the Open Source community is the ability to rapidly adapt to requirements changes as opposed to predicting all requirements in advance. Academic papers are littered with case studies of large software projects with extensive requirements planning that failed once the project was finished.[7,8,9] Many of these failures were due to requirements changing during the course of a lengthy development. Open Source does not suffer from this point of failure due to the iterative nature of Open Source development and the ability to quickly adapt to changing requirements.

One final comparison with agile methods is worth examining: Proponents of agile methods note that software development is as much people-oriented as process-oriented. Since Open Source development is the communion of developers, it is primarily people oriented (i.e., they would not be there unless they wanted to be).[10,19] But more to the point, the tools and processes for CSD were developed to fit the modes and temperaments of developers. Molding the methods to the people is the people-oriented aspect of CSD that yielded the greatest benefits.

VA Software was interested in learning more about CSD. The success of our original business model hinged on assuring the success of Open Source projects in general, and advancing the success of the Open Source community. We set out to understand how the Open Source community collaborated on development projects, why they were successful, and how the process could be improved.

Our investigations resulted in SourceForge.net, and ultimately SourceForge Enterprise Edition. This developer's portal for the Open Source community hosts more software projects than any other point on the Internet. The success of SourceForge.net is due in part to our review of the nature of CSD and how we enhanced Internet based collaboration.

When we designed SourceForge.net, we extended and enhanced the processes and tools that Open Source developers used to accelerate their efforts. The portal was designed to:

- Minimize administrative work
- Maximize communications and collaboration
- Preserve project knowledge
- Make it easy to establish projects and recruit experts
- Find and leverage existing code
- Do all of this on a global scale

4. ENTERPRISE DEVELOPMENT – AN OPEN SOURCE MICROCOSM?

Though enterprise development organizations often reflect many of the positive aspects of the Open Source community, there are some glaring differences. It is worth discussing these given that they can be corrected.

4.1 Mobility of resources

Larger enterprises have software development organizations that on many levels organizationally resemble the Open Source community. Developers are divided between teams based on areas of technical expertise. The largest difference between the Open Source community and development teams in enterprises is that enterprise developers do not get to choose the projects on which they work, much to their dismay.

And herein is one area where enterprise development organizational inflexibility hampers software development. The Open Source community thrives in part because developers are mobile resources. Open Source developers jump from project to project because no artificial barriers exist. There are no org charts, no segregation of responsibilities, no factional loyalties aside from technology prejudices. Developers are completely free to take on new challenges.

In enterprises this degree of project mobility is rarely facilitated. Large multi-national corporations with global development efforts often cannot make developers from one group or division available to others within the same company due to bureaucracy or ignorance of available talent. This limitation separates people with significant expertise from critical projects and more rapid software development.

Part of the problem is simply knowledge about available resources. One of the most active subsystems in SourceForge.net is the "Help Wanted" database. Within SourceForge.net, developers can list their areas of expertise as well as post their resumes. Project leaders can search for experts to work on their projects. It is rare for a multi-national enterprise to have this level of resource identification and recruitment potential.

4.2 Culture of sharing

Enterprise development organizations are not breeding grounds for collaborative efforts. Little is done by management – aside from code reuse campaigns and object library development – to promote the sharing of knowledge and code between developers.[11,12,15,16]

In the Open Source community, sharing of expertise and code is considered the norm and not the exception. Shared expertise is another explanation for the rapid ascension of Open Source as a viable source of tools. Any Open Source project manager can quickly find experts to collaborate – even on a short-term basis – on his or her project. Experts eliminate learning curves and help in avoiding novice errors.

Sharing of expertise cannot be forced. Developers need to want to contribute to other projects and the success of other developers. CSD depends on facilitating a common forum for experts to find one another, examine their works, and collaborate on projects or core technologies.

4.3 Peerage

Much of the collaborative nature within the Open Source community that is absent in enterprise development organizations revolves around peerage. Enterprise organizations often promote competition and not mutual support through standard individual performance reviews and bonus systems. Thus the peer processes common in the Open Source community do not naturally spawn in enterprise settings.

Peerage has two primary benefits in the Open Source world.

Peer review: Because code is developed in the SourceForge.net CSD environment, all products are available for review by the developer community at large. Discussion and critiques are common, rapid, boisterous, and contribute greatly to the development of stable and secure products (indeed, the Open Source peer process has been credited with creating software more immune to Internet threats than software produced by enterprises – a fact recently discovered by Microsoft).[13,14]

Peer glory: Developers thrive on peer approval. Public recognition for their products is more valuable to a developer than a bonus check. Within a CSD platform like SourceForge.net, developers have the opportunity to "show off" by having their work visible to their peers. This not only promotes participation in Open Source, but it also promotes collaboration because developers know the best way to avoiding embarrassing bugs and design flaws is to get early and frequent input from their peers. Enterprises could, but rarely do, allow their developers that momentary bit of glory that motivates them to work more creatively and effectively.

5. THE CHANGING ENTERPRISE DEVELOPMENT CYCLE

Interestingly, enterprises are experiencing many of the same software development obstacles as the Open Source community. The nature of enterprise software development is changing due to both technological and market forces.

When businesses began asking us for a commercial version of SourceForge.net, we started our investigations anew, focusing on how large organizations develop software now and how they want to develop software in the future. Combined with market research data, we were able to create a clear picture of the current state of enterprise software development and how it can evolve.

Some of the forces at work in enterprise software development are:

Remote developers: Enterprises are under demand to employ remote developers. Presently 61% of enterprises have some remote development and 50% outsource some or all of their software development [5]. Remote developers can be telecommuters (a competitive benefit afforded to many developers), consultants, developers from other divisions within a large enterprise, and offshore development companies. The need for hiring experts in specific technologies is accelerating the demand for outsourcing.

Rapid changes and iterative projects/processes: The rate of change in technology is growing as is the demand for software solutions that provide competitive advantages. Enterprises must deal with more new technology while accommodating internal demands for more strategic software capabilities.[17] Combined, they are forcing enterprises into a more iterative development

model resembling that of agile method advocates and the Open Source community.

Scattered and incomplete information: With developers coming from far flung corners of the world, development managers are facing new difficulties in capturing everything known about software projects, and making this knowledge reusable. But enterprises are keenly aware that knowledge capture can be a distraction from development work – enterprises clearly need non-intrusive tools for collecting intelligence, intellectual property, code and documentation.

Team communications: Development managers tell us that facilitating team communications was essential to their future success. The use of remote developers and the "soloist" nature of many developers complicate collaborative communications. Developers needed new ways of communicating about their projects and the opportunity to investigate other projects to expand their technical horizons.

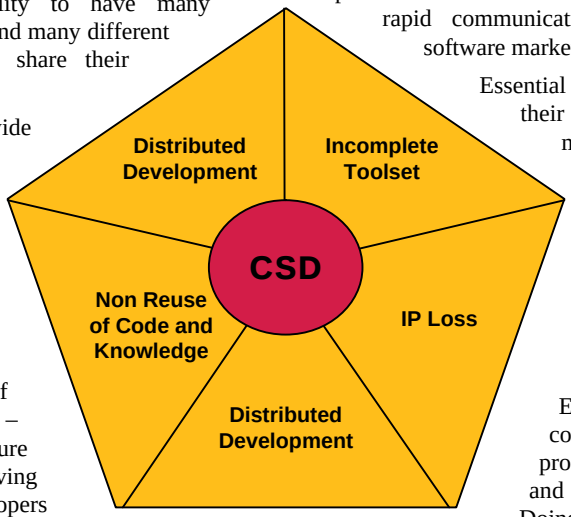
Code and knowledge reuse: Most development managers understand the value of reusing code. Mature code improves the quality of new projects and helps accelerate their completion. Most managers though do not have a pain-free way of making code available to widely distributed teams in a way that made finding reusable code practical. They almost universally lack simple tools for making the knowledge that went into projects available for new projects.

6. ENTERPRISE CSD PAIN POINTS

When these forces were compared to typical development teams, we discovered five "pain points" in enterprise software development. These pain points are places where software development typically fails to achieve organization goals. The five pain points are:

Distributed development: The ability to have many developers from many different teams and many different locations collaborate effectively and share their expertise.

Incomplete development tools: Provide tools that facilitate the management of code, knowledge, collaboration and projects, and integrate this knowledge to present a unified and consistent view of projects, code and knowledge.



Intellectual property loss: With high developer turnover rates and a growing remote developer base, a great deal of intellectual property and knowledge – that could benefit planning and future projects – is being lost. Preserving knowledge and expertise and developers rapidly shifting from project-to-project, and from team-to-team, is essential to accelerating future projects.

Duplicated coding efforts: Our investigations showed that a significant amount of new code duplicated functionality found in existing projects, resulting in unnecessary development spending and software release delays. Enterprises need to rapidly identify code and knowledge that can be reused without inducing cumbersome processes in capturing, classifying and indexing these resources. They also need the ability to identify experts on select technologies so they can "reuse" that expertise.

Excessive administration time: Developers and their managers alike are overwhelmed with the administrative effort required to keep projects on track. Developers in particular are resentful of how project management functions reduced their development time. The Open Source community lives – and indeed thrives – without cumbersome processes and administrative details. Management's mission is to remove administrivia from the lives of developers without sacrificing knowledge capture and project management.

With only nominal exceptions, this list of enterprise software development pain points matches the obstacles that the Open Source community overcame both through informal means, and through the SourceForge.net developer portal. Indeed, the Open Source community appears to be a model for enterprise development practices of the future.

7. CONCLUSIONS

Enterprises can learn a great deal from the trials and successes of the Open Source community. Open Source survives because it can harness the intellect and expertise of developers around the globe, permitting the best resources to be applied to each project. The Open Source community creatively uses the Internet in general, and SourceForge.net specifically, to overcome the barriers normally imposed by time and distance. Enterprises are beginning to shop the globe for experts-on-demand, following the Open Source model for project recruitment.

But more importantly, the Open Source community modeled their development practices and tools to achieve nearly impossible goals, including the development of an operating system that is gradually eliminating all but one proprietary operating system. Open Source's reliance on agility, iterative development, and rapid communications were crucial to changing the software market.

Essential to the Open Source success story is their highly collaborative nature. Self-managing teams of widely removed developers works because collaboration is the hub of all their activities. Management overhead and administrative burdens are nearly eliminated through frequent collaboration on each development phase and small, iterative changes to their products.

Enterprises leveraging software as competitive tools need to examine the processes of the Open Source community and rely on the same tools and practices. Doing so will accelerate enterprise development efforts and remove the pain of besting the competition.

8. REFERENCES

[1] B. P. Miller, L. Fredriksen, and B. So, "An Empirical Study of Unix Utilities," Communications of the ACM, 33, 12, Dec 1990, pp. 32-44. ftp://grilled.cs.wisc.edu/technical_papers/fuzz.ps.

[2] B. P. Miller, D. Koski, C. P. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl, “Fuzz Revisted: A Re-examination of the Reliability of Unix Utilities and Services,” Computer Sciences Department, University of Wisconsin-Madison, 1995. ftp://grilled.cs.wisc.edu/technical_papers/fuzz-revisted.ps.

[3] J. E. Forrester, and B. P. Miller, “An Empirical Study of the Robustness of Windows NT Applications Using Random Testing,” Computer Sciences Department, University of Wisconsin-Madison, Jul. 2000. ftp://grilled.cs.wisc.edu/technical_papers/fuzz-nt.ps.

[4] Martin Fowler, “The New Methodology,” Software Development magazine, December 2000, http://martinfowler.com/articles/newMethodology.html

[5] Evans Data, “Enterprise Development Management Issues,” December 2001

[6] Martin Fowler, Jim Highsmith, “The Agile Manifesto”, Software Development, August 2001, http://www.sdmagazine.com/documents/s=844/sdm0108a/

[7] Douglas C. Schmidt, Ralph E. Johnson, Mohamed Fayad, “Software Patterns”, Communications of the ACM, October 1996

[8] Robert L. Glass, “Software Runaways: Monumental Software Disasters”, Prentice Hall Inc, 1998, ISBN 0-13-673443-X

[9] Lawrence Bernstein and David Klappholz, “Teaching Old Software Dogs, Old Tricks”, Stevens Institute of Technology, http://www.cs.stevens-tech.edu/NJCSE/Papers/Papers_TeachingOldSoftwareDogs.htm

[10] E. S. Raymond, “The cathedral and the bazaar”, 11 November, 1998, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

[11] Eschenfelder, K., Heckman, R. and Sawyer, S., (1998) "The Distribution of Computing: Cooperation Among Distributed Technical Specialists," Information Technology & People, 11(2), 84-103.

[12] Sawyer, S. and Guinan, P., (1998) "Software Development: Processes and Performance," IBM Systems Journal, 37(4), 552-569

[13] Simson Garfinkel, “Security Through Obscurity”, WideOpen News, 12 November, 1999

[14] Pete Loshin, “Open Source Under The Hood “, Information Security Magazine, March 2001

[15] Dr. Will Tracz, “Did someone say reuse? Well excuse me!”, Flashline Newsletter, October 2001

[16] James Highsmith, “Adaptive Software Development: A Collaborative Approach to Managing Complex Systems,” Dorset House; ISBN: 0932633404

[17] Steve McConnell, “Rapid Development: Taming Wild Software Schedules”, Microsoft Press, ISBN: 1556159005

[18] Kent Beck, “Extreme Programming Explained: embrace change,” Addison Wesley, 2000, ISBN: 0201616416

[19] Greg Perkins, “Culture Clash and the Road to World Domination”, IEEE Software, January 1999

[20] Huaiqing Wang, Chen Wang, “Open Source Software Adoption: A Status Report”, IEEE Software, March 2001