

Open Collaboration within Corporations Using Software Forges

Dirk Riehle, John Ellenberger, Tamir Menahem, Boris Mikhailovski, Yuri Natchetoi, Barak Naveh, and Thomas Odenwald, SAP

Software forges are tool platforms that originated in the open source community. Many corporations are improving and extending their software development practices by adopting forges internally.

Over the past 10 years, open source software has become an important cornerstone of the software industry. Commercial users have adopted it in stand-alone applications, and software vendors are embedding it in products. Surprisingly then, from a commercial perspective, open source software is developed differently from how corporations typically develop software. Research into how open source works has been growing steadily.¹ One driver of such research is the desire to understand how commercial software development could benefit from open source best practices. Do some of these practices also work within corporations? If so, what are they, and how can we transfer them?

This article describes our experiences using open source software development practices at SAP. SAP is a major software developer and leader in business applications. We've found that open source practices can complement traditional top-down software development with bottom-up collective intelligence. Software forges offer a mechanism for advancing the adoption of open source best practices within corporations (see the sidebar, "What Is a Software Forge?"). We illustrate our experiences using SAP's own internal software forge, called SAP Forge, and compare our experiences with those from other large software companies.

Open Source Best Practices

Eric Raymond compared most corporate software development to building a cathedral—planning,

managing, and executing the carefully crafted work of skilled individuals from the top down.² In contrast, Raymond described open source development as a bazaar: no master plan, diverse agendas, and much redundant effort.

Many open source best practices fly in the face of traditional software development methods.^{3,4} For example, open source projects don't hide the source code from users; instead, they treat users as beta-testers. They also frequently release incomplete systems and, in general, don't view users as customers who expect a polished product. Rather, they empower users to become co-developers.⁵

A case study by Vijay Gurbani and his colleagues shows how companies can benefit from applying open source practices internally.⁶ Gurbani and his colleagues developed an internet telephony server at Lucent using an open source approach. Through multiple stages, the initial research project evolved into the backbone of multiple commercial products, all based on the same server software. Gurbani provided the server software as a shared internal asset, including the source code. Over time, several product groups contributed to the project, without any top-down companywide

project planning. The project followed the Linux development model of “benevolent dictator” with “trusted lieutenants.” The result was high-quality, broadly used software that met user expectations and could be easily customized to different needs.

At SAP, we wanted to use an open source approach to make research-to-product successes like Gurbani’s server software happen more often and smoothly. To achieve this, we first needed to understand open source better.

Principles of Open Collaboration

Open source is said to be based on meritocracy.⁷ We found this principle to be used as an umbrella term for these more specific principles of collaboration in open source projects:

- **Egalitarianism.** Everyone can contribute. Open source projects are accessible on the Internet, and the project community typically includes anyone who wants to help.
- **Meritocracy.** Contributions are judged transparently on the basis of their merits. All decisions are discussed publicly on mailing lists and can be looked up for reference.
- **Self-organization.** Typically, no defined process is imposed from the outside. The project community itself determines how to go about its work.

We call these the principles of open collaboration. They contrast starkly with how most corporations manage their internal software development processes:

- **Assigned jobs.** Top-down resource assignment determines who works on what project or which piece of software.
- **Status rather than merit.** A hierarchy of junior and senior developers and architects implies status and usually determines who has the final word in design and implementation decisions.
- **Imposed processes.** A process-definition department in the organization determines which software development process to follow, and it’s binding on all projects.

Perhaps the most important benefit of open collaboration is the emergent phenomenon of volunteer software developers who find and contribute to a project by their own choice.

Benefits of Internal Open Collaboration

Although the principles of open collaboration are hardly typical of traditional software development

What Is a Software Forge?

A software forge is an extensible Web-based platform that integrates best-of-breed software tools for collaborative software development. SourceForge (sourceforge.net) is the best-known example on the Internet, hosting the largest collection of open source projects of any forge. Other examples are BerliOS (www.berlios.de), Codehaus (codehaus.org), and Tigris (tigris.org).

A software forge has two main views:

- a project portfolio view that lets a developer browse and find projects, and
- a project view that provides the developer tools for working on a specific project.

Developers who navigate to a particular project will see a project-specific view, which typically has two parts:

- a listing of the different tools available for the project, and
- a view specific to a selected tool.

A good software forge supports the whole software development process from idea generation, project definition, and product management to configuration management, build support, and bug tracking. The forge integrates all the tools supporting these activities in one interface and makes navigating among them easy. All projects use the same tools, so developers can easily switch between projects.

Project forges differ from CASE tools in that their design centers on open collaboration, making it easy to find a project, read about it, understand it, and contribute as a volunteer.

organizations, they offer benefits that account for corporate interest in them:

- **Volunteers.** Even within traditional top-down structured software development organizations, projects can gather internal volunteer contributions.
- **Motivated contributors.** Volunteers choose projects according to their own interests rather than by assignment. The decision to contribute is theirs and gives them opportunities to gain reputation and visibility in the company beyond their assignments.
- **Better quality through quasi-public internal scrutiny.** When development within the corporation is open, developers typically feel an extra incentive to strive for high-quality contributions.
- **Broad expertise.** Because volunteers can join from across the organization, they can significantly broaden the expertise available to a project. This helps projects reach goals more quickly at higher quality. Specifically, it can help fix problems more quickly and either prevent mistakes or capture them earlier.
- **Broad support and buy-in.** With volunteers

Software forges make it as easy as possible to find a project, understand it, and contribute to it.

from across the organization, projects find a broader base and support in the organization.

- Better research-to-product transfer. Research projects can get expertise and volunteers from downstream product units, which can ease the research-to-product technology transfer.

At the root of these benefits are volunteer software developers. Researchers have studied public open source projects to determine how volunteers join them. For example, Georg von Krogh and his colleagues analyzed how volunteers joined the Freenet project;⁸ Israel Herraiz and his colleagues did the same for the Gnome project.⁹ These researchers found the joining process for volunteers to be gradual, compared to paid developers who undergo a rather abrupt, fully immersive experience.

Software Forges for Open Collaboration

Several large software vendors have taken steps to establish a consistent way of bringing open source best practices to corporate software development.

For example, Jamie Dinkelacker and his colleagues defined Hewlett-Packard's progressive open source program.¹⁰ As part of this program, they developed the "corporate source initiative," which supported the provision of HP Labs research projects as internal open source projects. Creating communities around these projects was key to their success. The communities consisted of not only researchers but also developers from product units.¹¹

IBM has a similar effort, which differs from HP's initiative in using an off-the-shelf software forge rather than custom-built software.¹² SAP also adopted this approach.

Gurbani's experience at Lucent showed that one of the biggest problems to internal open source is that many groups use different and frequently incompatible tools. A good software forge unifies the tool set and supports the whole software development cycle.

Forges and CASE Tools

In many ways, a software forge is like an integrated CASE tool.¹³ It provides a predefined but extensible set of tools that all play together to aid software developers in their project work. Task management, issue trackers, and documentation tools are common in both CASE tools and software forges.

The software development tools of many corporations are neither integrated nor complete, so

projects tend to install their own project-specific tools. Consequently, important project information is stored on different servers and is frequently lost as a project moves on in its life cycle. Software forges and CASE tools address these issues by giving developers one defined place with all the tools they need.

Thus, they both make economic sense. Among other benefits, they centralize and store important information and reduce resources spent on administrative tasks such as maintaining a project-specific Web server and bug tracker.

Critical Forge Design Issues

Corporate software developers are the primary market for CASE tools, along with the people who define the development processes. In contrast, software forges emerged on the Internet, and open source software developers are their primary users.

Open source projects tend to be resource-starved, so most projects must attract volunteers. Consequently, software forges are geared toward making it as easy as possible to find a project, understand it, and contribute to it.

Finding projects. A forge offers a product portfolio view first and a project-specific view second. Projects are indexed and searchable using one and only one URL as the starting point. Finding projects is easy on a forge and one of its most important features.

This contrasts with corporate software development, where a silo mentality frequently hides projects on separate servers, unindexed and with a cryptic URL. CASE tools are also typically project-centric and make a project visible only to the developers assigned to it.

Understanding projects. A typical forge offers project forums and mailing lists as a basic feature and makes the discussions on them accessible to the proper audience—by default, anyone who can access the forge. These discussions are how projects on the forge document their decisions and software development. Indexing and search mechanisms let users easily find which decisions the project made and why. This documentation approach makes it easier for volunteers to read about a project and get involved.

In contrast, discussions leading to decisions in traditional projects frequently occur in meetings or on the phone. They're sparsely documented, if at all. Frequently, this leaves developers with no more information about a decision than the deci-

Table 1**Data for the Hewlett-Packard, IBM, SAP, and Microsoft forges**

Corporation	Start date	No. of developers	No. of projects	At forge age (in months)
IBM ¹²	Jan. 2000	800 (~4% of population)	45	18
HP ¹⁰	June 2000	1,500 (~7.5% of population)	24	18
SAP	Sept. 2006	706 (~7.1% of population)	179	18
Microsoft	June 2007	794 (~2.8% of population)	406	10

sion itself. Discussions repeat themselves, and developers have much more difficulty getting up to speed on a project.

Contributing to projects. A forge offers developers tools that either they're already familiar with from previous work or they can familiarize themselves with quickly because the tools are used repeatedly across all forge projects. The first step to getting involved can be as easy as clicking on the reply button in a project's discussion forum and making a comment. This reduces the technical and practical hurdle of joining and becoming active in a project.

Although a well-run software development organization typically provides a defined tool set, we've found that many organizations have difficulty integrating them in a coherent offering. Developers therefore find different setups across different projects, even in the same company. This makes it hard to contribute quickly to a specific project, thus inhibiting volunteerism. In addition, traditional corporate projects tend to be defensive and hide their information, operating on a need-to-know basis rather than a desire to show themselves.

The SAP Forge Case Study

We're the leadership team for SAP Forge, which we designed with the benefits of open collaboration in mind. SAP's own software development process provides best-of-breed tooling. Since 2006, SAP Forge has enabled projects to acquire and keep volunteers within the corporate firewall's boundaries.

We based SAP Forge on the GForge (www.gforge.org) open source software forge code base. GForge is a popular choice in corporations; for example, IBM uses it, too.

SAP Forge represents one common platform found at one specific, easy-to-remember company-internal URL. Everyone within the corporation's firewalls can access it. Everyone who's interested can become a developer on the forge, and everyone can register a new project without going through

a lengthy approval process. Unless explicitly requested, all projects are open and accessible to everyone who cares to look.

SAP Forge has grown steadily since the company launched it in September 2006. Projects aren't required to use SAP Forge; it's a choice left to the project lead. One year after its inception, SAP Forge had reached more than 100 projects and had more than 500 registered users, representing about 5 percent of the overall SAP developer population. SAP Forge's overall growth has been linear, but we expect it to slow down once we've reached a sizable chunk of all SAP developers.

Table 1 compares SAP Forge with IBM, HP, and Microsoft forges. Steve Fox and Joe Latone of IBM provided the IBM data, and Andrew Begel provided the Microsoft data. SAP Forge has a substantially larger number of smaller projects than IBM and HP, which we attribute to a large influx of small research projects that were already complete and were looking for an easy-to-find resting place. We also believe that developers today are more comfortable with sharing code internally than they were a few years ago—also evident in the participation data from Microsoft, which launched its forge in 2007.

SAP Forge first gives developers an overview of all projects and developers on the forge. They can then switch to their dashboard, which shows them all projects they're involved in (see Figure 1 on the next page).

After the developer switches to a specific project, SAP Forge provides the expected tools such as bug tracking, configuration management, task management, forums, mailing lists, and wikis.

Project search, developer information, and project publicity were all important in introducing open source best practices to SAP. Users can search project names and descriptions to find those they're interested in. They can look at the profile and skill sets of developers on the forge and search for developers with specific skills. This supports and strengthens the emergence of a network of developers who know whom to turn to for advice

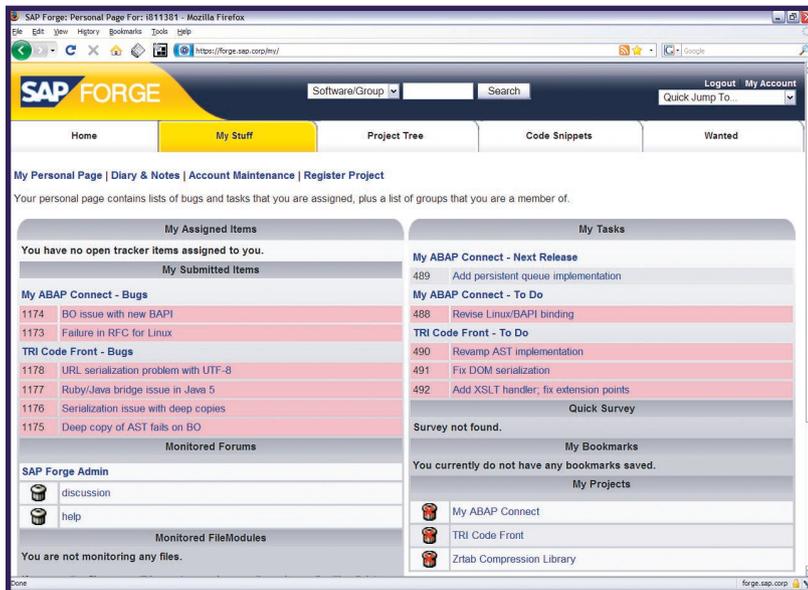


Figure 1. SAP Forge Developer Dashboard. Each developer on the forge has an individual dashboard that shows his or her currently active projects. (The data displayed here is fictional.)

and questions. And finally, SAP Forge gives users various general statistics, such as the most active or most popular projects. These statistics not only inform people but also motivate contributions because they imply recognition of the project and its developers. This in turn motivates higher-quality work.

An Example SAP Forge Project

One early SAP Forge project was the Mobile Retail Demo, a demonstration project that used the Bluetooth protocol for mobile shopping. The software lets mobile phone users configure what information they would like to receive on their phones from nearby stores—for example, information about an ongoing sale. In addition, users can voluntarily supply information about their tastes and current shopping list to retail shops. One project goal was to lay a foundation for a future Mobile Retail Framework to follow the demo project.

The project began in mid-2006 for demonstration at an SAP internal conference. In late 2006, it moved onto SAP Forge to get wider exposure in the company. The project was already a success, but moving it to SAP Forge drastically increased its reach and speed. The original development team consisted of three researchers. Fourteen months later, 27 developers had registered on SAP Forge for the project. Most new contributors were volunteers; there was no traditional top-down resource assignment.

The Mobile Retail Demo project leaders confirmed that the open collaboration more than achieved its reputed benefits. Specifically, the project experienced the following advantages from being on SAP Forge:

- **Volunteers.** SAP Forge brought the project more than 18 additional contributors. These contributors aren't full-time resources, but they do contribute actively and help the project move forward.
- **Motivated volunteers.** The volunteers joined the project on their own and hence care deeply about it, leading to contributions of above-average quality.
- **Broad expertise.** The volunteers brought expertise from across the organization. Many of them are working on related projects and are familiar with the problems of such applications.
- **Better understanding of requirements.** In addition to broad expertise, volunteers contribute insights into requirements and future applications that influence the project's product management.
- **Broad support.** The breadth of volunteers means broader support for the project across the organization. The project got such good publicity in the organization that further resources became available.
- **Testing help.** Enthusiastic volunteers who bought into the project became excellent human software testers, providing quick feedback on problems and bugs.
- **Increased visibility.** Being on SAP Forge and getting volunteers and broad support raised the project's profile and lowered the chances of redundant competing efforts, because everyone in the space knows about the project.
- **Formalized display of significance.** The contributions, broader interest, and raised project profile imply additional validation of its significance for the company.

The project leaders also expect an improved research-to-product transfer. The Mobile Retail Demo was a research project, but the volunteers it attracted included some of the more foresighted developers from product units. We expect this early buy-in from development to ease the technology transfer by aligning research interests and product needs early.

The project's exposure on SAP Forge and the forge's support for open collaboration have helped make the Mobile Retail Demo significantly more successful than would have been possible using only traditional management practices of corporate software development.

SAP Forge Benefits and Challenges

The Mobile Retail Demo was a big success for the SAP Forge, but it wasn't the only one. In a survey,

66 percent of all respondents (55 of 83) reported that they looked outside their silo, browsing for other projects that interested them, and 24 percent stated that their project received outside help, mostly bug reports and suggestions for improvement. Another 12 percent said they helped other projects on the basis of personal interest.

The managers of research projects are generally supportive of the volunteer contributions, as they expect to benefit from outside help and an improved research-to-product transfer process. The managers of volunteers from regular product units are typically skeptical in the beginning. We've found that they became neutral or even supportive once they realized the future benefits of early engagement with research projects.

The biggest hurdle to widespread adoption of SAP Forge is its limited compliance with tools mandatory for SAP's general software development process—in particular, SAP's proprietary Advanced Business Application Programming system. Initially, as a volunteer effort, we didn't have the resources to integrate the forge with these external tools. We're doing this now, expecting to draw even more projects to the forge. SAP Forge isn't in competition with existing tools and processes. Rather, it complements them, unifying existing standalone tools under one common user interface.

Although the SAP Forge tools enable developers to volunteer for projects they're interested in, we've found the open collaboration principles we described earlier to be crucial to retaining them. Projects must exhibit a mindset that welcomes whoever comes along to help rather than viewing volunteers as a foreign element (egalitarianism principle). Project members must realize that important input and contributions can come from across the organization and can be based on perspectives that might be unfamiliar to the original developers (meritocracy principle). Finally, SAP has well-defined software development processes, but accepting volunteer contributions sometimes means adjusting to volunteer needs and timelines (self-organization principle).

For volunteers, the main reward of the forge platform for open collaboration is their successful contributions to projects of their choice and the recognition they receive in doing so. We encourage project leaders to find simple ways to express their appreciation—for example, handing out project-specific T-shirts and talking to a volunteer's manager before a performance review.

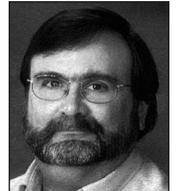
From an employer's perspective, an internal

About the Authors



Dirk Riehle is a senior research scientist at SAP Labs in Palo Alto, California, where he directs SAP's open source research efforts and leads SAP Forge. His research interests include all things relating to software engineering and collective intelligence. Riehle received his PhD in computer science from ETH Zurich (the Swiss Federal Institute of Technology). He's a senior member of the ACM and a member of the IEEE. Contact him at dirk@riehle.org.

John Ellenberger is a director of research at SAP Research in Boston. His primary research interest is the business application of emerging technologies, particularly open source, Web 2.0, and mobility. Ellenberger received his MS in computer science from Iowa State University. Contact him at johne@jellenberger.org.



Tamir Menahem is a development architect at SAP Labs in Ra'anana, Israel. His research interests include many aspects of software technologies and architectural concepts. Menahem received his MBA from Tel Aviv University with a specialization in technology and information systems. Contact him at tamir.menahem@sap.com.

Boris Mikhailovski is a chief IT architect at SAP Labs in Ra'anana, Israel. Contact him at bmikhailovski@sap.com.



Yuri Natchetoi is a senior research scientist at SAP in Montreal. His research interests focus on mobile business applications. Natchetoi received his PhD in computer science from Moscow Engineering Physics University. Contact him at yuri.natchetoi@sap.com.

Barak Naveh is the chief technology officer of Moblica, Israel. At the time he worked on the research reported here, he was with SAP. His research interests include software engineering and the practice of making really good software. Naveh received his MSc in computer science from Ben-Gurion University of the Negev. Contact him at barak@moblica.com.



Thomas Odenwald is head of ICW Technology Labs in San Mateo, California. At the time he worked on the research reported here, he was with SAP. His research interests include Web-based platforms and solutions that combine the key elements of the medical supply chain. Odenwald received his MS in computer science and economic engineering from Freidericana University Karlsruhe. Contact him at thomas.odenwald@icw.global.com.

software forge lets employees work on specific projects that interest them and helps avoid losing them to other activities. With an internal forge, we've found that enthusiastic developers with time, energy, and motivation are more likely to spend their effort for the good of the company than for outside projects. 

CALL FOR ARTICLES

The Cooperative and Human Aspects of Software Engineering

PUBLICATION: November/December 2009

SUBMISSION DEADLINE: 8 April 2009

This special issue focuses on multidisciplinary research and practice that explore how cooperative and human issues affect software development and evolution, both in terms of challenges and successes. Accepted articles must address either cooperative or human aspects as they relate to software engineering and must offer practical, reliable insights that can be applied in real-world software development contexts. Case studies, experience reports, and articles about empirical studies, lab studies, novel tools, or novel processes are welcome.

THIS SPECIAL ISSUE SEEKS ARTICLES ON

- Software engineering as cooperative work
- Industrial experience reports examining the influence of CHASE in software projects, such as the influence of personality, leadership, or effective teamwork
- Social and cultural aspects of software engineering
- Psychological and cognitive aspects of software engineering
- Coordination in large-scale software development
- Cooperation between software developers and other professionals over a system's lifetime

GUEST EDITORS:

- Janice Singer, National Research Council Canada, janice.singer@nrc-cnrc.gc.ca
- Li-Te Cheng, IBM
- Cleidson de Souza, Federal University of Para, Brazil
- Gina Venolia, Microsoft
- Helen Sharp, The Open University, London

FULL CALL FOR PAPERS:

www.computer.org/software/cfp6.htm

FOR AUTHOR GUIDELINES AND SUBMISSION DETAILS:

software@computer.org
or www.computer.org/software/author.htm.

References

1. W. Scacchi, "Free/Open Source Software Development: Recent Research Results and Emerging Opportunities," *Proc. 6th Joint Meeting European Software Eng. Conf. and the ACM SIGSOFT Symp. Foundations of Software Eng.* (ESEC/FSE 07), ACM Press, 2007, pp. 459–468.
2. E. Raymond, *The Cathedral and the Bazaar*, O'Reilly, 2001.
3. C. DiBona, S. Ockman, and M. Stone, *Open Sources: Voices from the Open Source Revolution*, O'Reilly, 1999.
4. K. Fogel, *Producing Open Source Software*, O'Reilly, 2005.
5. E. von Hippel, *Democratizing Innovation*, MIT Press, 2005.
6. V.K. Gurbani, A. Garvert, and J.D. Herbsleb, "A Case Study of a Corporate Open Source Development Model," *Proc. 28th Int'l Conf. Software Eng.* (ICSE 06), ACM Press, 2006, pp. 472–481.
7. K.R. Lakhani and R.G. Wolf, "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," in *Perspectives on Free and Open Source Software*, MIT Press, 2005, pp. 3–22.
8. G. von Krogh, S. Spaeth, and K.R. Lakhani, "Community, Joining, and Specialization in Open Source Software Innovation: A Case Study," *Research Policy*, vol. 32, 2003, pp. 1217–1241.
9. I. Herraiz et al., "The Processes of Joining in Global Distributed Software Projects," *Proc. 2006 Int'l Workshop Global Software Development for the Practitioner*, ACM Press, 2006, pp. 27–33.
10. J. Dinkelacker et al., "Progressive Open Source," *Proc. 24th Int'l Conf. Software Eng.* (ICSE 02), ACM Press, 2002, pp. 177–184.
11. C. Melian et al., *Building Networks of Software Communities in a Large Corporation*, tech. report, HP Labs, 2002.
12. D. Sabbah, "The Open Internet—Open Source, Open Standards and the Effects on Collaborative Software Development," presentation at the 2005 High Performance Transaction Systems Workshop, 2005, www.hpts.ws/papers/2005/agenda.html.
13. S. Jarzabek and Riri Huang, "The Case for User-Centered CASE Tools," *Comm. ACM*, vol. 41, no. 8, 1998, pp. 93–99.

IEEE
Software

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.