(Accepted for publication in IEEE Software, pending professional editing)

Bringing Open Source Best Practices into Corporations using a Software Forge

Dirk Riehle[1], John Ellenberger[2], Tamir Menahem[3],
Boris Mikhailovski[3], Yuri Natchetoi[2], Barak Naveh[3], Thomas Odenwald[1]

dirk@riehle.org, {firstname.lastname@sap.com},
b.mikhailovski@sap.com, barak {at sign} moblica {dot} com, thomasodenwald@yahoo.com

ABSTRACT

A software forge is a tools platform for collaborative software development, similar to integrated CASE environments. Unlike CASE tools, however, software forges have been designed for the software development practices of the open source community. In this paper, we discuss our experiences with using a software forge to bring open source best practices into corporations. We present the design principles and benefits of a firm-internal software forge, and we present a case study of how one project inside SAP benefitted significantly from being on the forge.

Categories and Subject Descriptors: D.2 Software Engineering, D.2.6 Programming Environments, D.2.9 Management, H.5 Information Interfaces and Presentations, H.1.2 User/Machine Systems, H.5.3 Group and Organization Interfaces, K.6 Management of Computing and Information Systems, K.6.1 Project and People Management, K.6.3 Software Management.

Keywords: software forge, collaborative software development, open source, open source best practices, open collaboration.

1 INTRODUCTION

Over the last 10 years, open source software has become an important cornerstone of the software industry. Commercial users use open source software as stand-alone applications and software vendors embed it as components into their products.

Surprisingly then, from a commercial perspective, open source software is being developed in ways that are different from the way corporations typically develop software. Research into open source best practices illustrates the desire to understand how open source software development works [1]. One of the drivers of such research is to understand how commercial software development could benefit from open source best practices. Are there best practices that work within corporations also? If so, what are they, and how can they be transferred?

[1] SAP Research, SAP Labs LLC, 3475 Deer Creek Rd, 94304 Palo Alto, CA, U.S.A

[2] SAP Research, SAP Canada, 111 Rue Duke, Montréal, Quebec H3C, Canada

[3] SAP Labs Israel, 15 Hatidhar St. P.O. Box 4077, 43665 Ra'anana, Israel

This article describes our experiences with using open source software development practices within SAP. We have found that open source practices can complement traditional software development with bottom-up collective intelligence which brings volunteers and expertise to projects that otherwise may not have received such. We then discuss software forges as a specific mechanism for furthering the adoption of open source best practices within corporations. We illustrate our experiences using SAP's own internal software forge, called SAP Forge, and by comparing them with experiences gathered at other companies like HP, IBM, and Microsoft.

2 OPEN SOURCE BEST PRACTICES

According to Raymond, most corporations develop software as if they were building a cathedral: It is carefully crafted work by skilled individuals with top-down management, planning, and execution [4]. In contrast to this, Raymond considers open source software to be like a bazaar: No master-plan, a divergence of agendas, and much redundant effort.

Many open source best practices fly in the face of traditional software development methods [5], [6], [7]. Open source projects, for example, do not hide the source code from their users; rather they treat users as beta-testers; also, they frequently release incomplete systems. Open source projects generally don't view users as customers that expect a polished product, but rather they empower users to become co-developers [16].

A case study by Gurbani and colleagues shows how companies can benefit from applying open source practices internally [15]. Gurbani 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. The server software was provided as a shared asset, including source code. Over time, several internal product groups contributed to the project, without any top-down company-wide project planning. The project followed the model of the "benevolent dictator" with "trusted lieutenants" as explored by Linux. The result was broadly used high-quality software that met their users' expectations and could easily be customized to different needs.

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

2.1 Principles of open collaboration

Open source is said to be based on the principle of meritocracy [8]. We have found that the principle of meritocracy is used as an umbrella term for the following three more specific principles of open source:

- *Egalitarian.* Everyone can contribute, because open source projects are accessible on the Internet and the project community is typically inclusive to anyone who wants to help.
- *Meritocratic.* Contributions are judged transparently and based on their merits. All decisions are discussed publicly on mailing lists and can be looked up for reference.
- *Self-organizing.* There is typically no defined process imposed from the outside so the project community itself determines how to go about its work.

We call these principles the principles of open (source) collaboration. These principles are in stark contrast to 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 the software. Thus, volunteer contributions to other projects are virtually unheard of.
- *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 within the organization determines which software development process to follow and it is binding to all projects.

Perhaps the most important benefit of the three principles of open collaboration is that they lead to the emergence of the phenomenon of volunteer software developers who find and contribute to a project by their own choice.

2.2 Benefits of "being open" internally

Many benefits accrue from the three principles of open collaboration. These benefits are typically not found within traditional software development organizations. However, like others [2], [6], [9], [15], we have found that we can bring some of them to corporate software development:

- *Volunteers.* Even within traditional top-down structured software development organizations, projects can gather internal volunteer contributions.
- *Motivated contributors.* Volunteers choose projects based on their own motivation rather than by getting assigned. Volunteers are motivated to contribute, because it is their decision to contribute and they can gain reputation and visibility within the company outside their current primary projects.
- *Better quality through quasi-public scrutiny.* With development being open (within the corporation), developers typically feel compelled to strive for high quality of contributions.
- *Broad expertise.* Since volunteers can join from all across the organization, the expertise brought to a project broadens significantly. Such increased breadth of expertise has the benefits of reaching goals faster at higher quality. Specifically, broader expertise can fix problems faster and prevents mistakes or captures them earlier.
- *Broad support and buy-in.* With volunteers from all across the organization, projects find a broader base and support within the organization.
- *Better research-to-product transfer.* Research projects can get expertise and volunteers from downstream product units. Such early buy-in from the product units eases the research-to-product-unit transfer.

At the root of these benefits are volunteer software developers. For public open source projects, von Krogh et al. analyze how volunteers joined the Freenet project [10]. Herraiz et al. discuss how volunteers join the GNOME project [11]. They compare the joining process between volunteers and paid developers and conclude that volunteers undergo a gradual process while paid developers have a rather abrupt and fully immersive experience.

2.3 A consistent approach to firm-internal open source

We are not the first to realize the benefits of a consistent approach to bringing open source best practices to corporate software development.

Dinkelacker and colleagues, for example, defined Hewlett Packard's progressive open source program [2]. 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. Key was the creation of communities around these projects, consisting not only of researchers, but also of developers from product units [3]. The benefits HP expected from its corporate open source initiative were improved competitiveness, higher quality products, and lower costs.

IBM started a similar effort at around the same time [17] [19]. While HP's corporate source initiative used custom-built software, IBM used an off-the-shelf *software forge* to realize its initiative. This is also what SAP did.

3 SOFTWARE FORGES

A software forge is an extensible web-based platform that integrates best-of-breed software tools for collaborative software development. A forge hosts projects, all accessible through one common URL, and provides each project with the necessary tools.

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

Please see the sidebar *Appendix A: What is a Software Forge?* for a short introduction to software forges.

3.1 Forges and CASE tools

In many ways, a software forge is like a modern integrated CASE environment [12], [13]. It provides a pre-defined 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.

Many corporations today are in a situation where their software development tools are neither integrated nor complete so that each project installs its own additional tools. Consequently, important project information is stored on different servers and is frequently lost as a project moves on in its life-cycle.

Thus, both software forges and CASE tools make economic sense. Among other benefits, they centralize and store important information and reduce resources spent on administration efforts like maintaining a project-specific web server and bug tracker.

3.2 Critical forge design issues

CASE tools, whose primary audience is corporate software developers (and the people who define their software development processes), are typically found only in corporations. Software forges, in contrast, emerged on the "open Internet" and have open source software developers as their primary audience.

Because open source projects tend to be resource-starved, the dominant need of most projects is to attract volunteers. As a consequence, software forges are geared towards making it as easy as possible to find a project, understand it, join it, and contribute to it.

The following aspects of a software forge are central to the process of getting and keeping volunteers:

- *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 made easy and is an important feature of the forge.

  This is in contrast to corporate software development, where silo mentality frequently hides projects on their own servers, not indexed, and with a cryptic URL that makes them hard to find. CASE tools typically have a project centric view and make a project only visible to a developer if he or she has been assigned to it.

- *Reading-up on projects (documented history).* A typical forge offers projects forums and mailing lists as a basic feature and makes discussions on them accessible to the proper audience, by default to anyone who can access the forge. Thus, software development on the forge documents its project discussions and decisions. Indexing and search mechanisms make it easy to find which and why decisions were made. Thus, documentation makes it easier for volunteers to read up on a project and get involved.

  In contrast to this, discussions leading up to decisions in traditional projects are frequently done in meetings or on the phone and hence get sparsely documented, if documented at all. Frequently, this leaves developers with no more information about a decision than the decision itself. Discussions are bound to repeat themselves, and it is much harder for a developer to get up to speed with a project.

- *Getting involved.* A forge offers tools that a developer is already familiar with (because they are used to it from previous work) or quickly gets familiar with (because the tools are repeatedly used across all projects on the forge). 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. Such setup reduces the technical and practical hurdle of joining and becoming active in a project.

  While a well-run software development organization typically provides a defined tool set, we have found that many organizations find it hard to make them a coherent integrated offering. Thus, developers find different

setups across different projects, even within the same company, which makes it hard to contribute quickly to a specific project, preventing volunteers. In addition, traditional corporate projects tend to be defensive and hide their information, asking for a "need to see" rather than having a desire to show themselves.

We have designed a software forge with these goals and properties in mind and have found that they significantly aid the process of getting and keeping volunteers, even within the boundaries of the corporate firewall. These volunteers then provided the benefits discussed above like volunteer contributions, broader and deeper expertise, and support and buy-in across the organization.

4 SAP FORGE CASE STUDY

SAP is a major software developer, the world's leader in the overall space of business applications. SAP's own software development process provides best of breed tooling, and since 2006, has included a thriving internal software forge called SAP Forge. The authors of this article are the leadership team of SAP Forge.

4.1 History and technology

SAP Forge is based on the open source software forge (code base) of GForge [14]. GForge is used within corporations, for example, at IBM and SAP, and for engaging with partners and customers, for example, at ThoughtWorks and SugarCRM.

SAP Forge represents one common platform found at one specific and memorable company-internal URL. It is accessible to everyone within the firewalls of the corporation. Everyone who is interested can become a developer on the forge and everyone can register a new project without having to go through a lengthy approval process.

SAP Forge was launched in September 2006 and has been growing steadily since then. Choosing SAP Forge for a project is not mandatory but 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% of the overall SAP developer population. The overall growth of SAP forge has been linear, but we expect it to peter out once we have reached a sizable chunk of all developers.

Table 1 compares SAP Forge with other forges. SAP Forge has a substantially larger number of smaller projects, which we attribute to a large influx of smaller frequently already finished research projects. Also, and this is evident in the Microsoft data as well, we believe that developers today are more comfortable with sharing code internally than they were 6-8 years ago.

	Start Date	# of developers	# of projects	at age of forge [mths]
IBM [17] [19]	January 2000	~800 (~4% of population)	45	18
HP [2]	June 2000	1500 (~7.5% of population)	24	18
SAP	September 2006	706 (~7.1% of population)	179	18
Microsoft [18]	June 2007	794 (~2.8% of population)	406	10

Table 1: Data for the HP, IBM, and SAP forges after 10 or 18 months of operation

SAP Forge first provides an overview of all projects and developers on the forge. Developers can then switch to their dashboard which shows them all projects they are involved in. Figure 1 shows a screenshot.

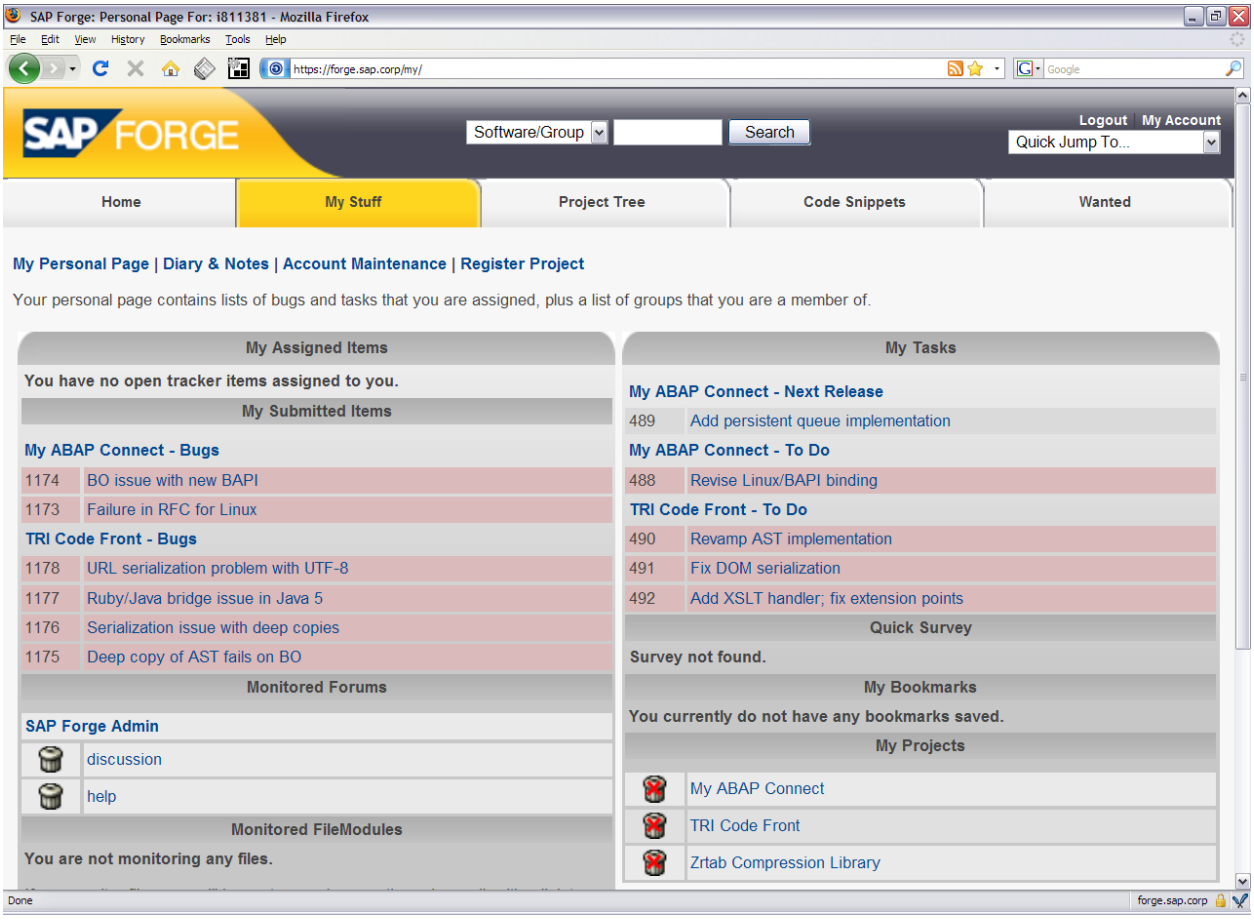

Figure 1: A developer dashboard in SAP Forge (original screenshot with mock-up data)

After switching to a specific project, SAP Forge provides the to-be-expected tools like bug tracker, configuration management, task management, forums and mailing lists, wikis. It looks similar to Figure 3 in the Sidebar.

From the perspective of introducing open source best practices into SAP, important features are:

- *Project search.* Users can search the project names and project descriptions to find projects they may be interested in. Unless explicitly requested, all projects are open and accessible to everyone who cares to look.
- *Developer information.* Users 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 who to turn to for advice and questions.
- *Project publicity.* Users are given various statistics like what are the most active or most popular projects. These statistics not only inform people, they also motivate to contribute as such statistics imply recognition of the project and its developers. This in turn motivates higher quality of work.

4.2 Mobile Retail Demo example

An early project on SAP Forge was the Mobile Retail Demo project, a demonstration project for mobile shopping.

The software lets users of a mobile phone configure what information they would like to receive on their cell phone from nearby stores using the Bluetooth protocol. Such information could be about an on-going sale. In addition, the user might be willing to supply to the retail shops information about his or her tastes and current shopping list. One goal of the project was to lay a foundation for a future Mobile Retail Framework to follow-up on the demo project.

The project was started in mid 2006 to be first shown at an SAP internal conference. In late 2006 it moved onto SAP Forge to get wider exposure within the company. It already had been a success, but moving it to SAP Forge drastically increased its reach and speed.

When the project first appeared on the forge, development had already moved the software to a usable state. The original team consisted of three researchers. 14 months later, the number of registered developers on the forge for the project had grown from 3 to 27. Most of the new contributors are volunteers who decided by themselves to help the project; there was no traditional top-down resource assignment.

The project leads of the Mobile Retail Demo project confirmed that the benefits of open collaboration were reached well beyond expectations. 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 are not full-time resources but they do contribute actively and help the project move forward.
- *Motivated volunteers.* The volunteers decided by themselves to join the project and hence care deeply about the project, leading to above-average quality contributions.
- *Broad expertise.* The volunteers bring broad expertise from all across the organization to the project. Many of the volunteers 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 product management of the project.
- *Broad support.* The breadth of volunteers also means a broader support for the project across the organization. The project got such good publicity within the organization that further resources became available.
- *Testing help.* Enthusiastic volunteers who bought into the project care deeply about its quality and hence are the best possible (human) testers of the software, 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, as everyone in the space knows about the project.
- *Formalized display of significance.* Also, the contributions, broader interest, and raised profile of the project imply additional validation of its significance for the company.

While being realized only now, the project leads are also expecting an improved research-to-product transfer: The Mobile Retail Demo project is a research project that gathered some of the more foresighted developers from product units as volunteers. We expect this early buy-in from development into research to ease a later transfer, as interests and needs are aligned early.

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

4.3 SAP Forge benefits and challenges

The Mobile Retail demo was a big success, but not the only one. In a survey, 66% of all respondents (55 of 83) reported that they looked outside their silo, browsing for other projects of interest to them. 24% of all respondents stated that their project received outside help, mostly in the form of bug reports and suggestions for improvement. And another 12% of all respondents said they helped out other projects based on 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 have found that they became neutral or even supportive once they realized the future benefits of early engagement with research projects.

The biggest hurdle to wide-spread adoption of SAP Forge is its limited compliance with tools mandatory for SAP's general software development process, in particular SAP's proprietary ABAP system. Initially, as a volunteer effort, we did not have the resources to integrate the forge with these external tools. We are now doing this, expecting to draw even more projects to the forge. SAP Forge is not in competition with existing tools and processes but rather

complements them: It unifies existing stand-alone tools under one common user interface and it enhances the traditional top down resource allocation process with bottom-up collective intelligence.

The tools on SAP Forge make it possible for developers with time, energy, and motivation to find, join, and contribute to projects they are interested in. To make this happen, however, the tools need to be used wisely. We have found that the open collaboration principles given earlier were crucial to making volunteers happen. In our communications, we emphasize the corresponding mindset:

- *Egalitarian.* Project members need to be inclusive of whoever comes along to help rather than viewing them as a foreign element. Documenting the project with future readers in mind is a core aspect of this.
- *Meritocratic.* Project members need to realize that important input and contributions can come from all across the organization based on perspectives that may be unfamiliar to the original developers.
- *Self-organizing.* SAP has well-defined software development processes. Still, projects need to be accommodating of their volunteers and respectful of their time.

For volunteers, successful contribution and positive feedback from the project is the main reward in itself. Nevertheless, we encourage project leads to find creative ways of expressing their appreciation. Simple ways of doing so are handing out project-specific T-shirts or talking to a volunteer's manager before a performance review.

From an employer's perspective, an internal software forge lets employees contribute to projects that interest them and helps avoid loosing them to other activities. With an internal forge, we have found that enthusiastic developers are more likely to spend this additional effort for the good of the company rather than for outside projects.

5 CONCLUSIONS

Corporations can benefit significantly from bringing selected open source best practices in-house. Specifically, corporations can complement the traditional top-down resource allocation process in software development with a bottom-up process in which interested parties self-select and contribute as volunteers. Such volunteers are frequently effective contributors because bring expertise and support from all across the organization. We have found corporation-internal software forges to be an excellent tool for enabling such volunteers. We present the design principles of SAP's internal software forge and we illustrate the benefits using a three-person project that gathered an additional 18 volunteers over the course of 14 months.

REFERENCES

[1] Walt Scacchi. "Free/Open Source Software Development: Recent Research Results and Emerging Opportunities." In *Proceedings of ESEC/FSE 2007.* ACM Press, 2007.

[2] Jamie Dinkelacker, Pankaj K. Garg, Rob Miller, Dean Nelson. "Progressive Open Source." In *Proceedings of the 24th International Conference on Software Engineering* (ICSE 2002). ACM Press, 2002. 177-184.

[3] Catharina Melian, Cathy Burles Ammirati, Pankaj Garg, Guje Sevon. "Building Networks of Software Communities in a Large Corporation." *HP Labs Technical Report.* Hewlett-Packard, 2002.

[4] Eric Raymond. *The Cathedral and the Bazaar.* O'Reilly, 2001.

[5] Chris DiBona, Sam Ockman, Mark Stone. *Open Sources: Voices from the Open Source Revolution.* O'Reilly, 1999.

[6] Karl Fogel. *Producing Open Source Software.* O'Reilly, 2005.

[7] Ron Goldman and Richard P. Gabriel. *Innovation Happens Elsewhere.* Morgan Kaufmann, 2005.

[8] Karim R. Lakhani, 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. 3-22.

[9] Dan Woods, Gautam Guliani. *Open Source for the Enterprise*. O’Reilly, 2005.

[10] Georg von Krogh, Sebastian Spaeth, Karim R. Lakhani. “Community, Joining, and Specialization in Open Source Software Innovation: a Case Study.” *Research Policy* 32 (2003). 1217–1241.

[11] Israel Herraiz, Gregorio Robles, Juan José Amor, Teófilo Romera, Jesús M. González Barahona. “The Processes of Joining in Global Distributed Software Projects.” In Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner. ACM Press, 2006. 27-33.

[12] Stan Jarzabek, Riri Huang. “The Case for User-Centered CASE Tools.” Communications of the ACM 41, 8 (August 1998). 93-99.

[13] Iris Vessey, Ajay Paul Sravanapudi. “CASE Tools as Collaborative Support Technologies.” Communications of the ACM 38, 1 (January 1995). 83-95.

[14] GForgeGroup. GForge Collaborative Development Environment. See http://www.gforge.org.

[15] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. “A Case Study of a Corporate Open Source Development Model.” In Proceedings of the 28th International Conference on Software Engineering (ICSE 2006). ACM Press, 2006. 472-481.

[16] Eric von Hippel. Democratizing Innovation. The MIT Press, 2005.

[17] Steve Fox, Joseph Latone. Personal Communication. IBM, 2008.

[18] Andrew Begel. Personal communication. Microsoft, 2008.

[19] Danny Sabah. “The Open Internet – Open Source, Open Standards and the Effects on Collaborative Software Development.” Presentation at the 2005 High Performance Transaction Systems Workshop. Pacific Grove, CA: 2005.

APPENDIX A (SIDEBAR): 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. The best-known example of a software forge on the Internet is SourceForge, which hosts the largest collection of open source projects of any of the forges. Other examples are Berlios, Codehaus, and Tigris.

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 a developer with tools to work on a specific project. Figure 2 shows the project overview page of SourceForge. As you can see, you can (A) search for projects, (B) browse by categories, and (C) find interesting projects by chance.

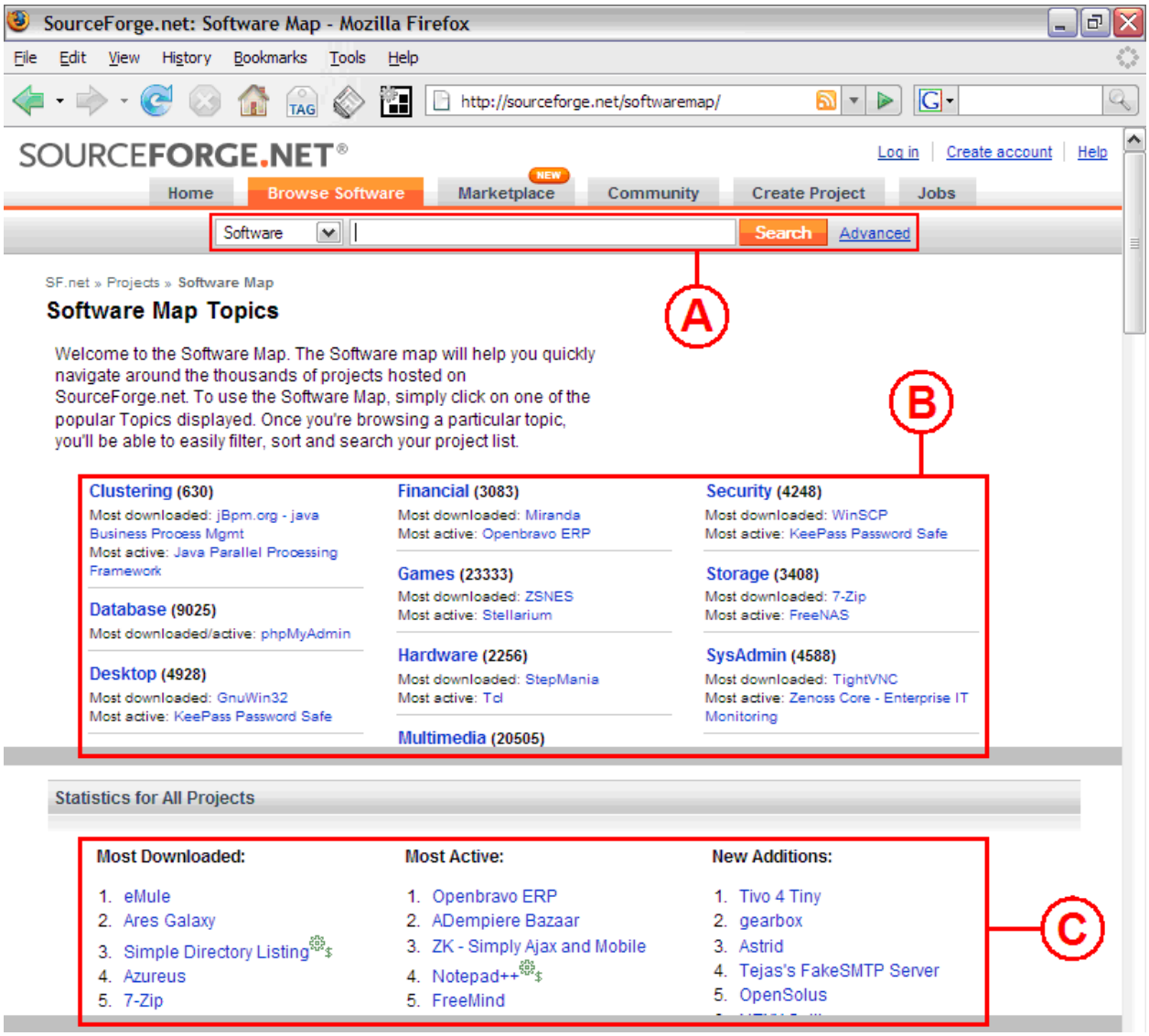

Figure 2: Project overview on SourceForge (advertisements removed, screenshot shortened).

If a developer navigates to a particular project, he or she will see a project-specific view, which typically has two parts: The top part lists the different tools available for the project, and the bottom part shows a view specific to the tool selected. In Figure 3, taken from the Berlios software forge, the bug tracker of the J2ME Polish project is being displayed. You can see tools like Forums (discussion forum), Bugs (bug tracker), Features (product management), Tasks (task management), a wiki, and many more.

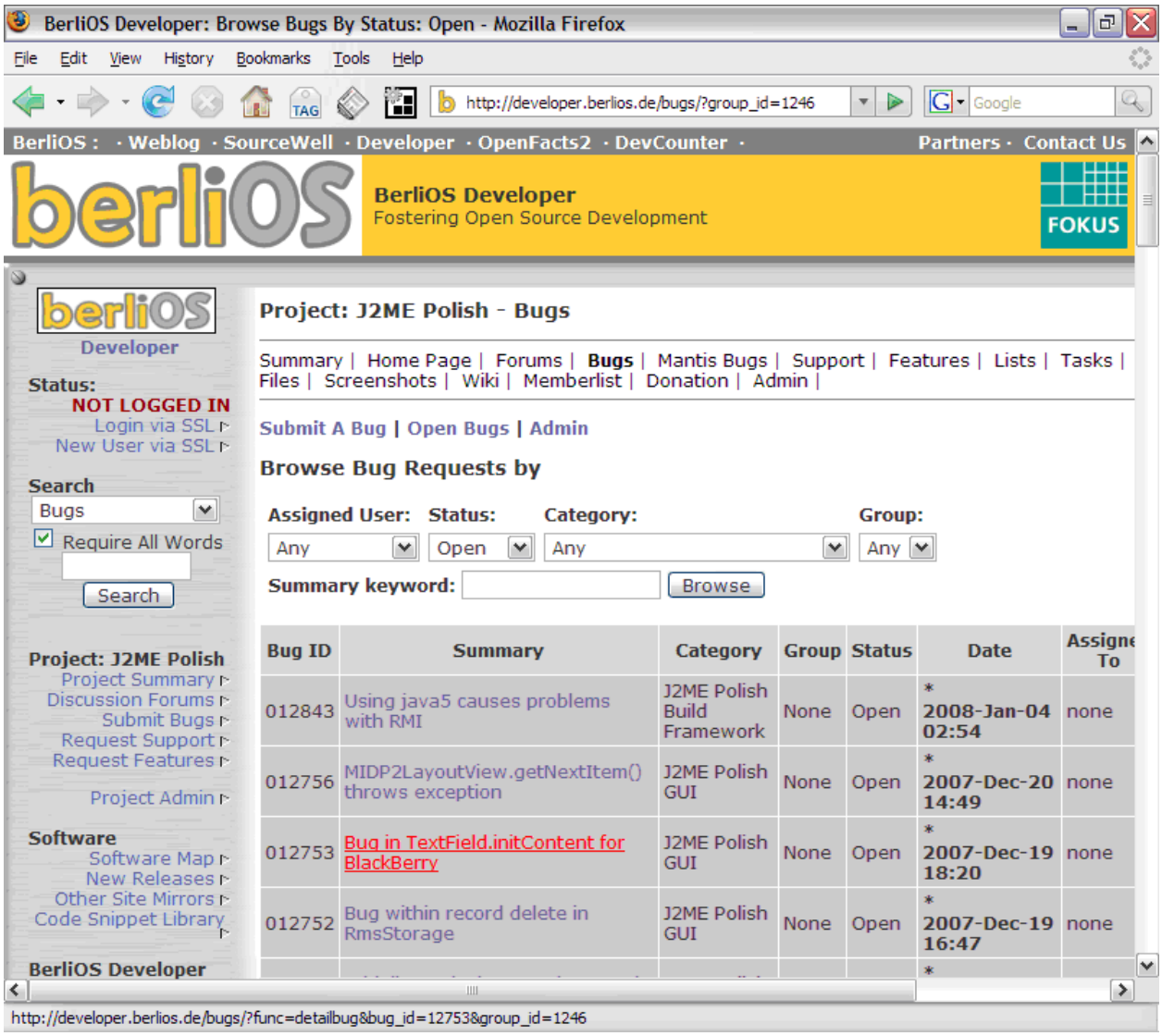

Figure 3: Project view of J2ME Polish on the Berlios forge (advertisements removed, screenshot cleaned up)

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. All the tools that support these activities are integrated with each other and it is easy to navigate between them. All the tools are the same between different projects so that it is easy for developers to switch between projects.

Like CASE tools, software forges make economic sense: Project managers and developers get a complete tool set at the click of a button and don't have to worry about technical administration of the tools, as this is typically provided by an IT organization rather than the project itself. Unlike CASE tools, software forges have been designed for open access that makes it easy to find a project, read-up on and understand it, and contribute to it as a volunteer. This article discusses the benefits of such open collaboration.