Progressive Open Source

Jamie Dinkelacker, Pankaj K. Garg, Rob Miller, and Dean Nelson

Hewlett-Packard Company
Palo Alto, CA 94303
(650)857-4709
garg@hpl.hp.com

ABSTRACT

The success of several Open Source™ software systems, e.g., Apache, Bind, Emacs, and Linux, has recently sparked interest in studying and emulating the software engineering principles underlying this innovative development and use model. Certain aspects of the Open Source development method, e.g., community building, open discussions for requirements and features, and evolvable and modular designs are having fundamental and far reaching consequences on general software engineering practices.

To leverage such Open Source methods and tools, we have defined an innovative software engineering paradigm for large corporations: *Progressive Open Source (POS)*. POS leverages the power of Open Source methods and tools for large corporations in a progressive manner: starting from completely within the corporation, to include partner businesses, and eventually complete Open Source. In this paper we present the design goals and principles for POS. We illustrate POS with two programs in HP: Corporate Source and the Collaborative Development Program (CDP). We present early results from both these programs suggesting the power and necessity of POS for all modern large corporations.

1. INTRODUCTION

Software engineering continues to be challenging for large corporations, often due to the realities of corporate structure, and its constraints and processes. Within the corporate shell such pressures as "time to market" can lead to specific architecture or coding approaches that put speed at the forefront and can compromise features or quality. In a large corporation (upwards of 50,000 employees) with several large divisions (upwards of 1000 employees), coding standards may differ if they exist at all, and the divisions may be developing similar modules (e.g., encryption, decryption libraries) unaware that others in the company are working on the same task, or may have already finished them.

Large corporations are also prone to "re-orgs" where suddenly a set of developers are moved from one project team to another, or a product that the team was working on is either hastened to market or put on the back-burner. Either case leads to a disruption in the software development process for the developers. Similarly, corporate priorities can change due to market conditions, often leading to the reallocation of personnel to different positions and projects. As such, developers face steep learning curves resulting from the misalignment of experience, skills, and new tasks.

Importantly, such challenges, while related to software development, are not limited to corporations that sell software. They are endemic to all organizations that build software–to support their hardware products, to manage their IT infrastructure system, to provide specific services, or to sell software. Whether a company's software efforts are externally or internally focused, whether they are products themselves or embedded parts of a broader solution, the above challenges of software development are faced by all such organizations. *Our working hypothesis is that certain software engineering principles from the Open Source paradigm can mitigate such large organization challenges.*

Open Source has become a familiar concept for software development. Sparked by such notable successes as the World Wide Web (HTML and HTTP), the Linux operating system, the Apache web server, and the Emacs text editor, the value of the Open Source concepts, perspectives, methodologies, and approach have received widespread support from many developers worldwide. Open Source has created a "sea change" regarding how software is distributed, as well as how it is crafted. As the name implies, Open source reflects a fundamental shift from software being distributed in binary form to a source-code form. With source-code, developers and users can review it, comment on it, and where necessary, correct it. Usually such source-code is maintained in open repositories on the Internet where new developers can familiarize themselves and learn new techniques. As a result, "source" distributions have led to the creation of diffuse communities of practice surrounding the code. Similarly, within an Open Source process, discussions about features, requirements, and architecture are held in open public forums (usually news groups and mailing lists), thereby creating an ideal negotiation grounds for developers, users, and maintainers.

The Open Source software development process has proven itself by delivering several industrial strength software products. In each of these cases, Open Source resulted in substantial benefits for easier software maintenance, code reusability, and higher quality [14]. Often, these Open Source prod-

ICSE'02, May 19-25, 2002, Orlando, Florida, USA.

ucts are preferred by the market than their "Closed Source" counterparts. While the Open Source development method has various reasons for its success, one main ingredient was best summed up by Eric Raymond as "Linus' Law": *given enough eyeballs, all bugs are shallow.*

From a software engineering perspective, this can be viewed as a natural progression of the code inspections process [6]. Unlike code inspection, however, the Open Source process is much broader and pervasive in that even the software user community is routinely involved in critiquing and reviewing source code, as well as having strong input on features and approaches to implementation. In short, Open Source software has a tendency to match to the needs of the users as seen by developers from a broad range of perspectives, not just a few from one small development group of product and marketing managers.

Much like the evolution of scientific concepts and theories [11], Open Source encourages the development of engineering techniques for software in an open, peer-review process. Similar to the scientific process, the primary motivation for people involved in the process is a quest for knowledge and peer recognition [12].

1.1 Applying Open Source concepts to corporations

While attractive from a purely quality perspective as well as from the standpoint of including innovation, the Open Source mechanism creates a fundamental conflict for software corporations: business practices dictate that software corporations retain Intellectual Property (IP) rights in their software, hide such IP from their competitors, and make profit on their investment in creating such IP. Realizing this business conflict, several Open Source derivatives have developed over the past few years. Such corporate derivatives are usually more restrictive than the typical Open Source licenses. Another approach is for corporations to expose some part of their products as Open Source while keeping the critical ones as proprietary source. An extreme form of this, which is the norm in the software industry, is to reveal only the user interface or application programming interface (API's) for software. When software corporations adopt such proprietary source-code development processes, they lose potential benefits of a possible Open Source counterpart. These benefits have the potential to be broad-ranging and cover the gamut not only of finding shallow bugs, but speeding up time-to-market, improved software reuse, and rapid re-deployment of skilled developers.

1.2 Progressive Open Source

Working in a large corporation (upwards of 80,000 employees, worldwide presence), with a significant range and diversity of software development (including embedded systems, operating systems, networking, applications, mobile code, and systems management), we have been investigating the use and benefits of the Open Source development methods for large corporations. Keeping in mind that all such large corporations are sensitive about protecting commercial interests in the software they develop, we have designed a three-tiered model, called: **Progressive Open Source (POS)**, for introducing Open Source concepts in HP's development practices. The three tiers of POS are: (1) Inner Source, (2) Controlled Source, and (3) Open Source. Figure 1 shows the relative positioning of the three layers of this model.

Inner Source refers to the application of the Open Source approach and benefits to developers within the corporate environment – i.e., "open" to all developers behind the firewall. The overarching purpose of the Inner Source approach is to bring the benefits of development according an Open Source model inside a company to its community of developers, and then foster the development of internal projects accordingly. Although the community size is smaller than that of the Internet, a worldwide company does have many developers who can be served by this approach.

Controlled Source which is outside of the corporate firewall, but restricts access limited to specific corporate partners. Such partnerships include development partners, technology partners, testing partners, localization partners, and so forth.

Open Source refers to the "open" use of the Internet for development, and release of the software source code in a license approved by the Open Source Initiative (OSI) (`http://www.opensource.org`). The OSI has currently approved about two dozen such licenses.

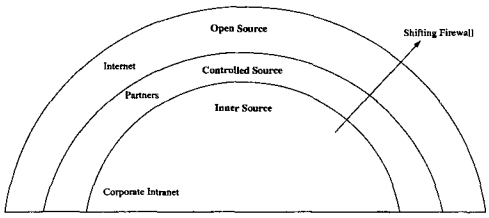

Figure 1: Progressive Open Source

POS requires a novel approach for large corporate software development: Instead of relying on a single-product, project-focused development method, POS requires a corporation-focused development method. With this method, each employee of the corporation can potentially contribute to the development of any given software product. Figure 2 depicts this organizational transformation. By restricting the openness of the software development to within the corporation, or with selected partners, the corporation does not incur the aforementioned business costs of Open Source. If the corporation is large enough (with a few hundred employees), it can realize the main benefit of Open Source, i.e., "given enough eyeballs, all bugs are shallow."

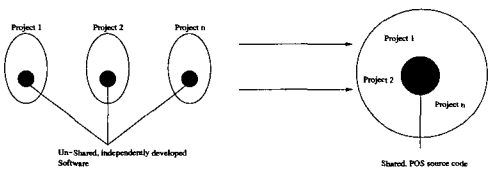

Figure 2: Transforming corporate Software Engineering

1.3 Benefits for Rapid Team Re-deployment

Corporate work is typically done by teams. In every organization, product shifts and reorganizations are common-

place. This subsequently creates a challenge as to how to effectively re-deploy skilled software developers, and once they're deployed, how to get them up the learning curve as quickly as possible. The start-up time of any team is significant. In an environment where time to market is always crucial, and having code that can be sustained to meet the needs of the marketplace and key customers is critical, the POS approach affords a new opportunity for having much quicker startup times for projects. Assuming an implementation of POS, familiar developers within the POS environment would already know the source tree, be familiar with the workings of the relevant development, bug-reporting, and source management tools, corporate-specific coding and commenting standards, and the corporate code review process. In short, POS creates an opportunity for rapid re-deployment of developers not just from one project to another *but from one* **product** *to another*. From the standpoint of managing software development in a commercial environment, this is a huge potential benefit.

1.4 Potentials for Partner Involvement

As the world becomes more and more characterized by evolving alliances and joint ventures, developers continually face the challenge of working together with those outside of of their corporation. Through appropriate access control and network security, either parts of the source tree can be placed outside the firewall, or within limits, partners can be allowed to tunnel into parts of it. Those aspects that a company chooses to share can be available to partners under non-disclosure agreements (NDAs) and work on joint projects can progress. While it is not uncommon for "Closed Source" vendors to establish specific working relationships with partners under strict NDAs, the code that is reviewed by the partners tends to be very focused on a specific product release or update. Also, this approach to code development is not standardized and is idiosyncratic to each company as well as to each product release. POS, based on Open Source methods, promotes standardization amongst the tools and techniques used in partner collaboration, which has the potential of increasing speed to market. The POS approach affords the opportunity for partners to work more broadly across the code tree, especially regarding foundation level modules (e.g., network stacks) and use more standardized approaches to coding and inspection that otherwise would be used with Closed Source approaches.

In summary, POS enables the following benefits for an organization:

- A readily available potpourri of software that can be built upon and used as starting point – provides for reuse, stability, and reliable code;
- Improved quality levels of shared software as authors' reputations are at stake;
- Shared, community debugging;
- Ability to easily integrate the corporate software development efforts into the overall Open Source movement, leverage the Open Source tools and methods, and ensure effective cross-learning where appropriate;
- Rapid re-deployment of key developers from one project to another who already are familiar with the current POS code tree, tools, and coding standards; and,
- Faster development schedules with code leveraged among several products;
- Opportunity to work with selected partners in certain circumstances.

2. CHALLENGES

Implementing POS within corporations has been–and will continue to be–challenging. Some of these challenges are organizational: e.g., how do we develop code across project and organizational boundaries, how do we identify and retain module designers? Yet other challenges are related to infrastructure technology. Both these aspects must be addressed in a satisfactory manner to ensure a successful and continuing deployment of POS. In this section we present some of the challenges of deploying and running a POS program in a large corporation.

2.1 Organizational

Organizational challenges of POS include:

Virtual Organizations: Most corporations today operate on a hierarchical organizational structure. This complicates the process of code sharing by having differing product road-maps and time-lines, where some managers may just push to get something delivered by a promised date, irrespective of code quality, which might then be an embarrassment to post into the POS code tree. It's also possible that some managers or even developers may be inimical to contributing any resources to perceived resource competitors within the organization.

Leadership: The Open Source model depends on at least one leader or owner of given software modules. Such leaders are efficient designers and implementers for the software module they are leading. In the open market of the Internet, "invisible hand" sorts of mechanisms (based on visibility, ego, and so forth) ensure that a good leader emerges for a given software module. In the case of POS, two distinct challenges arise: (1) what happens when a leader of a software module decides to leave the company, and several projects are critically depending on that module for their projects; or, (2) worse yet, no particular leader emerges for any given module?

Task Assignments: The Open Source model depends on a willing and able cadre of capable software engineers who work on any given software module. The pool of people to draw from is the entire world population! (As programming is getting easier, the world of programmers is increasing.) For Inner and Controlled Source, the pool of programmers to draw from is limited to the corporation and its partners. Traditionally, project managers can determine their personnel skill requirements based on the requirements of their projects. In the POS model, however, the entire company's pool of programmers can potentially help out in the development of a given project's source code. How does one manage the appropriate software skill set at the corporate level?

Developer Indoctrination: A fundamental aspect of software development is the skill set of each individual developer. This necessitates that corporate developers be aware of the POS tree and tools, that they adopt the coding standards set forth to be consistent with the source tree, and that they develop a good judgment as to what constitutes a reasonable contribution to the source tree. These are organizational and managerial challenges to broad adoption

and continued usage of POS within the enterprise. While many corporations have established and clear coding standards, this model can break down across divisions that serve distinctly different markets. This creates the challenge of maintaining coding standards, and training new developers for maximally utilizing POS.

2.2 Technology Infrastructure

The technology infrastructure challenges for POS include:

- **Security:** Corporate project managers and developers are wary of random exposure to the internals of their products, i.e., source code and design. Typically they want to at least know who is accessing their source code and for what purpose. This varies with projects from very loose security requirements to highly-sensitive company confidential information. Hence, POS requires appropriate authentication, authorization and audit mechanisms to properly control access to source code. This is contrast to the Open Source mechanism where potentially everyone has at least read access to source code.

- **Search and Navigation:** Once the POS repository attains a certain size, searching and navigating through the several projects and personnel details can easily become time consuming to the point of making the effort worthless. Hence, a proper search and navigation infrastructure is essential to the success of POS.

- **Migration from existing tools and Infrastructure:** Each software project and group within corporations today has a software engineering infrastructure in place which typically includes a version management system, debugging and bug reporting tools, and so forth. POS assumes a uniform toolset and infrastructure, at least from the user interface perspective. Migrating existing source code to a common infrastructure or user interface is a challenge, both from a technology and organization perspective.

- **Appropriate IT Support:** POS, like any server-centric content, requires hosting and maintenance of the code tree, platform, version control and related software engineering tools. While often overlooked, the IT support is absolutely crucial for maintaining uptime, running scheduled backups and recovery when necessary, and hardware maintenance as well.

3. APPROACH

To test out our ideas about POS, we have defined two programs in the Hewlett-Packard Company. The programs are: (1) **Corporate Source Initiative** championed within the HP Labs community, and (2) **Collaborative Development Program (CDP)** championed within HP's Imaging and Printing Business. Corporate Source is an instance of the *Inner Source* aspects of POS, while CDP is an instance of the *Controlled Source* aspects. In this section we discuss the approaches and vision of each program.

3.1 Corporate Source

3.1.1 Corporate Source Objectives

The purpose of the Corporate Source Initiative is to extend the HP Labs research community out into the product generation R&D community. HP Labs develops a wide variety of technology and prototypes, which HP has traditional difficulty transferring into the product generation portion of a business. Corporate Source, by utilizing the community focus of the Open Source principles, is working to build a stronger community within HP Labs and then extend that community to include product generation and other developers focused on infrastructure and corporate operations. This results in much greater leverage of HP Labs technology into HP's products and operations. The expected results are greater competitiveness of HP products, higher quality products, better return on HP Labs investment, decreased product generation costs, and a more effective and efficient enterprise.

Figure 3 shows the main web page for accessing the Corporate Source service.

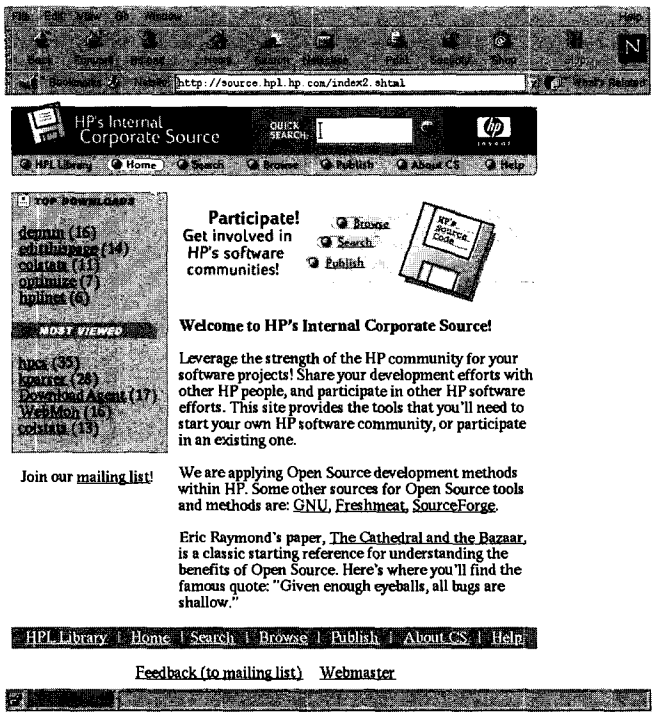

Figure 3: Main page for Corporate Source

HP Corporate Source is currently hosted through the HP Labs Research Library. The library pages are familiar to HP software engineers and thereby combines together in a known place for "corporate knowledge management." Appropriate IT functions are provided for hosting, system maintenance, and backup.

The Corporate Source web-site is searchable, and shows quick summaries, with active links, of the most frequent downloads, and the most frequently viewed modules. It provides an easy mechanism to upload modules. The HELP feature shows briefly how to use the site, whom to contact, and displays publishing guidelines (i.e., coding standards). Each submitted software module or item has a unique, one word software ID. Each software item lists one contact person, who is also the publisher of the software, although a

software item can have multiple authors. The source code is stored in a configuration management system, called Concurrent Version System (CVS) [9]. The information about source code is stored in XML files. Both these reside on a Linux Corporate Source server. Also, a mail-reflector is used to keep developers apprised of Corporate Source topics and submissions.

It's worth noting that not all HP developers are focused on generating software products – the company uses advanced decision technologies and supply chain management software as well. Much of this software is internally developed and won't necessarily be released into a product for obvious reasons related to competitive corporate performance. Yet, Corporate Source is well suited for long-term evolution of these tools and capabilities. In other words, Corporate Source is beneficial for a broad array of software developers within the company: developers in the Labs, in Infrastructure, Functions, as well as product generation groups.

3.1.2 Corporate Source Results

The Corporate Source service has been operational for more than fourteen months in HP. We have about two dozen software systems currently hosted by the service. Each of these systems was volunteered by the submittor. Figure 4 shows the average number of requests per day plotted against the month of operation. As the figure shows, recent months are averaging about 200 requests per day.

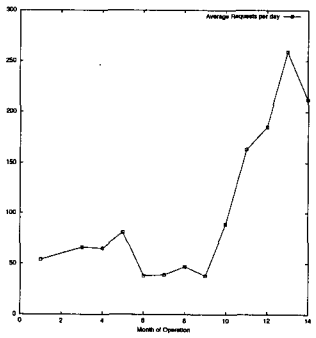

Figure 4: Request statistics for Corporate Source

3.2 Collaborative Development Program (CDP)

3.2.1 CDP Objectives

The purpose of the Collaborative Development Program (CDP) is to enable development teams to collaborate effectively on a global scale, with partners both within HP and external to the company.

CDP provides a comprehensive program to support HP's shift from a vertical organization focus to a virtual organization focus with:

- a supportive environment for virtual relationships.
- a set of supportive values and beliefs to better foster these relationships.
- a capabilities review and training program to ensure that HP's talent is enhanced to exploit this opportunity.
- a structure for ongoing behavioral migration to address these opportunities

The expected results of CDP include great agility to address new opportunities, reduced time to market, higher quality, more consistency between HP products, and decreased product generation costs.

3.2.2 CDP Program Structure

The CDP team consists of a set of executive champions, sponsors, R&D change leaders, and Information Technology staff.

Executive Champions: It has been critical to CDP success to have a group of executive champions. In CDP's case we have two champions - the chief technology officer and the chief information officer. These champions help provide the urgency to the organization to start the change process. They have provided both vision and direct influence open individuals perspective for the broad need of collaboration.

Sponsors: CDP has a group of nine sponsors. These sponsors were specifically chosen for their current business requirement for broad collaboration. The program ensured that each of these sponsors could feel the business pain of not having a collaborative model of operation. This pain was quantified by requiring that each sponsor commit resources to help CDP and a project to be hosted on the program. This helped ensure that they committed to a positive result.

R&D Change Leaders: The program was originally led by a group of R&D change leaders who would be the ultimate consumers of the program once successful. This was important for credibility and focus purposes. While IT is working to become aligned with direct business results, R&D productivity and flexibility is a bit too nebulous to quantify. In addition, IT's cost-focused history leaves them lacking the credibility to launch an R&D change management program.

IT Staff: IT experience with operations and support is absolutely critical to success. CDP, as a central R&D facility, can have a huge negative impact on productivity, which directly impacts both the top and bottom line returns. Maximum up time, especially when considering the global scale, is warranted.

3.2.3 CDP Offering

The CDP tools are currently hosted on the Internet outside of the HP firewall. This decision was made for a few reasons. First, HP's focus on "anywhere, anytime, always on infrastructure" warrants an Internet-based solution. The Internet allows people to work anywhere, any time and for all participants to be first class citizens in the development community independent of whether they are HP employees, contractors, or solution partners. This even opens the door for the direct inclusion of customers into the development, which is being explored on a limited basis. The second reason for hosting the solution on the Internet is the speed and agility in which we can bring on new partners. It previously required up to six months to establish an infrastructure to support a partner integration. With an Internet-based solution new partners can be added *within a day*. The third advantage of Internet hosting is the ability of contracting best in class hosting services for performance, reliability, and economies of scale.

The CDP toolset is integrated and hosted by an outside vendor. The vendor's framework integrates various open

source and non-open source tools together into a common integrated view of all project artifacts. HP has worked with the vendor to integrate HP's existing internal security system into their framework, to ease the internal migration of users to the platform. Figure 5 shows an example screen image of a CDP home page.

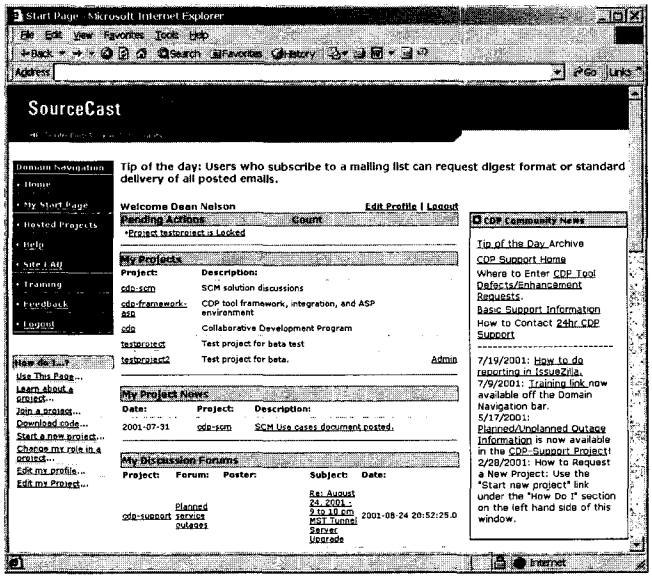

Figure 5: Home page for CDP. SourceCast is a TradeMark of CollabNet, Inc.

Realizing that the Open Source methods have more to do with community development and interaction than with the tools, HP has an internal culture team which is working to develop the operating models, educate our internal users and partners, and recognize and reward collaborative behavior. A collaboration skills assessment, training program, and exercises to experience this collaboration are all components of this initiative.

HP continues to evolve its intellectual property in this area and work with its partners to deliver a long-term road-map delivering deeper open collaboration.

3.2.4 CDP Results

The CDP team has completed a 6-month pilot program, which grew to over 600 users. This pilot completed in April 2001 as the program migrated to a production version. As of the end of August, CDP is hosting over 350 projects and 2000 users. Of those 2000 users approximately 10% are non-HP users. Figure 6 shows the weekly growth of new projects in CDP since April 2001.

There have been many examples of expected and unexpected returns from this more open collaborative approach. In addition, a number of key learnings have been noted. These learnings are discussed in the next section.

4. LESSONS LEARNED

Both Corporate Source and CDP have been in some level of pilot and production for nearly a year. During that period we have learned a great deal about not only changing the development processes of a major technology company but

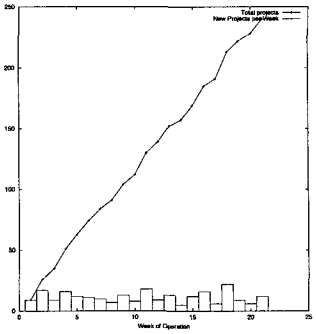

Figure 6: Weekly growth of projects in CDP

about changing the development culture as well.

Some learnings from our experiences are:

Adoption of Open Source practices within a corporation is, at its heart, more of a process of social–rather than technical–change. While it does require learning of new tools (e.g., CVS, mail reflectors, other web based communication methods), the key elements are: (1) leadership to draw people into using Open Source methodologies, and (2) getting developers and their management chain comfortable with sharing code and development responsibility across the company.

The motivations for participation in a corporate collaborative development environment can differ immensely from the motivations of the Open Source community. A high level of importance should be placed on identifying a value proposition that aligns the Open Source style of project development with business objectives. The rate of adoption and participation will be maximized when participation is critical to development needs as well as business and project success.

Training on topics ranging from tool use to community development and social change are necessary for all participants, including both developers and managers. As was noted earlier, the change embodied in moving to a collaborative development model is both social and technical: care should be taken to ensure that proper training exists and is encouraged in both areas.

The value of having the correct effective Champions and Sponsors should not be underestimated. In order to for this type of development to be successful key sponsors and leaders should be identified in both the management chain and in the development community.

It's a challenge to find an effective starting point regarding which software domain should be an initial focus from which to populate the repository. CDP approached this problem by tying participation to critical unmet business needs. In the case of CDP, those needs were identified as secure collaboration with third party partners and supporting geographically distributed teams. This was critical to the adoption of the tools, and methodologies as these needs were not being effectively met using existing tools and processes. This seeded the repository with resources that were critical to both the business and collaboration need. As our economies become more global these needs will continue to grow. After a time we expect a level of critical mass to occur so that the repository will be self-sustaining in regards to sharing and

leverage.

Using the word "module" somewhat generically, it's a challenge to strike the balance between module simplicity and utility. This age-old problem resurfaces with corporate collaboration because there's a tradeoff between how complete a module might need to be for another developer to consider adopting it into his or her code stream. Each submission should not have irrelevant features, APIs, or calling sequences that would take more time to reconfigure the module than write it from scratch. Effective heuristics for this are developing within POS in an evolutionary way.

Once a development community is established and is members understand the technical and social aspects of participation are understood community tend to be self-managing. For the early stages of adoption and in those rare cases where communities cannot come to a consensus a method of problem resolution and arbitration should exist that can enable problems and disputes to be resolved efficiently.

In the corporate development environment the protection of intellectual properties (IP) will continue to be an ongoing issue. Tools must be put into place that allow for broad sharing and collaboration while protecting valuable IP from being exposed inappropriately. This is particularly true when working with third party partners.

The growing need for global collaboration combined with the easy access to tools, has led to the use of these tools and methodologies in areas other than source code development. While this could lead to additional tool requirements, it presents opportunities for additional exposure and acceptance. It also presents opportunities for collaboration between disciplines that were not possible in the past.

5. RELATED WORK

The concept of Progressive Open Source embodies a fundamental shift in software development both from a product and process viewpoint. From a process viewpoint, the shift is from standard role definitions in broadening the population of possible developers to include other kinds of process agents: users, customers, partners, marketing, technical writers, and so forth. From a product viewpoint, the fundamental shift is that instead of the binary code as being the end-product, the source code is the end-product. Such fundamental shifts naturally have relationships and implications for several threads of software engineering research. We highlight such implications in this section.

The software reuse community has long espoused the notions of interface-exposing component-oriented software reuse. The theory is that as long as the interface to a component remains unchanged, its implementation can be changed to accommodate new developments or situations. In fact, for such reasons, Parnas and others have argued that the source code for the components should remain hidden from the users of components and only the interface be exposed [13]. This 'information hiding,' however, has a strong limitation: *an interface can never completely specify its implementation.* This is a natural consequence of the fact that the implementation is more than its interface, hence along with hiding the implementation details, the interface also hides the *hidden assumptions* in the implementation. While specifications of such assumptions can theoretically solve this issue, in practice specifications tend to lag behind the implementations. Hence, usage of the interface that are consistent with the assumptions work well, while novel or in-consistent usages break. This leads to significant debugging problems for component users.

The POS solution for this problem is to open the source code for the implementation while insisting on the use of only the interface. The users of such interfaces and the developers can then have meaningful conversations in an "Open Source" manner to refine or appropriately use the interfaces.

Several research groups are exploring the issues of of "Global Software Development (GSD)" within large software organizations, e.g., Lucent Technologies [8, 7], Alcatel [5], Motorola [2], Tenovis GmbH & Company [10]. They assume the work conducted in a hierarchical,
product-oriented organizational structures. Not surprisingly, Herbsleb *et al.* [8] report that communication and coordination over long distances hinder progress of code improvement or defect removals. Comparatively, Open Source projects report remarkable communication and coordination over long distances, e.g, see [12]. Our approach of Progressive Open Source blends the requirements of GSD with Open Source practices. For example, Herbsleb and colleagues list locating the person with the right expertise as one of the challenges of GSD [8][page 89]. Within POS, each participating person has a digital presence through their contributions to the mailing lists, news groups, or source trees. Locating the right person, therefore, is easily and naturally facilitated.

Abrahamsson [1] explores the role and definition of commitment in software process improvement programs.
While that paper focused on software process improvement, the ideas and lessons are relevant for the software process itself. For example, one of his conclusions for practitioners of software process improvement is "try to get people volunteering in [Software Process Improvement] SPI activities" (page 72). The Open Source software process fundamentally relies on *volunteers.* While we do not know of any scientific study of commitment in Open Source software development methods, the literature is abound with examples of heros with strong commitments towards the Open Source software they created, e.g., Linus Trovalds to Linux, Richard Stallman to Emacs, and Larry Wall to PERL [12]. We believe that similar levels of commitments can be created within organizations, with the type of organizational structure and infrastructure offered by POS.

Biffl *et al.* [3] describe positive results of re-inspecting software modules for defects. As we mentioned in the introduction section, Open Source extends the notions of inspections from a discrete, usually one or two-step process, to a *continuous* process. While not as structured as the code-inspection process, this continuous review process can potentially lead to better quality code. For example, Raymond [14] cites several anecdotal cases where defects were detected and resolved quickly because source code was made available to users and other developers. POS provides a fundamental infrastructure for such *continuous inspection* processes.

Boehm's risk-driven spiral model of software development [4] has found common use in industrial practice. The model dictates that software development process should follow a spiral from requirements through prototyping and development, with the most risky aspects of the software being addressed in the inner most loops of the spiral. This way, the aspects of the software that are the riskiest are addressed first, thereby reducing the chances of project failure. The Open Source model seems to follow a spiral, although the

spiral is driven by *useful features* [12, 14] rather than by the risk. It's a common practice in Open Source software development to address the immediate feature requirement of a user, often by the user himself. Such features (requirements through implementation) are then immediately tried out by other users, critiqued and improved upon. Further studies of POS will determine to what extent such Open Source *feature-driven* spirals are used in the corporate environment.

Software architectures have been studied in the past few years in the software engineering literature for several properties of reuse, extensibility, performance, and so forth. Open Source projects typically have a strong up-front architecture design [9], usually from an expert designer. One common theme in Open Source project architectures is that the architectures are *open for extensions*, i.e., new developers (users) can add their own extensions and features quite easily. This is true of a variety of successful Open Source projects like Apache, TEX, PERL, Linux, Emacs, to name a few. While it's too early to define the correct architectures for POS, programs like CDP and Corporate Source will help determine such architectures.

6. SUMMARY

We have defined the Progressive Open Source (POS) model for large corporations to take advantage of the Open Source development model. POS embodies the Open Source concepts of opening up the source code of software for review and modifications from others, and building communities around software development. Unlike Open Source, however, POS does not directly pose a threat to the competitive advantage of corporations in embedding novel engineering and algorithms in their software system.

To take advantage of the POS method we have defined two large programs within HP: (1) Collaborative Development Program, and (2) Corporate Source. Both these programs have been in use for over six months and we showed some usage results from these programs that demonstrate an increased usage of these programs in HP and a stabilization of the tools. Security classification and enforcement is emerging as an important issue. We have developed organizational plans for the success of the two programs. In the future we anticipate to refine the organizational and cultural structure for the continued success of the programs.

Availability

The Corporate Source software is available for download from
ftp://src.hpl.external.hp.com/pub/open/hpcs/

Acknowledgements

We acknowledge the work of several people in HP and several HP partners, too many to name here, who have made Corporate Source and CDP possible and successful.

7. REFERENCES

[1] Pekka Abrahamsson. Committment Development in Software Process Improvement: Critical Misconceptions. In *Proceedings of the* 23^{rd} *International Confernce on Software Engineering*, pages 71–80, Toronto, Canada, May 2001.

[2] Rober D. Battin, Ron Crocker, Joe Kreidler, and K. Subramanian. Leveraging Resources in Global Software Development. *IEEE Software*, 18(2):70–77, March/April 2001.

[3] Stefan Biffl, Bernd Freimut, and Oliver Laitenberger. Investigating the Cost-Effectiveness of Reinspections in Software Development. In *Proceedings of the* 23^{rd} *International Confernce on Software Engineering*, pages 155–164, Toronto, Canada, May 2001.

[4] Barry W. Boehm. A Spiral Model of Software Development and Enhancement. In Richard H. Thayer, editor, *IEEE Tutorial: Software Engineering Project Management*, pages 128–142. IEEE Computer Society Press, 1988.

[5] Christof Ebert and Phillip De Neve. Surviving Global Software Development. *IEEE Software*, 18(2):62–69, March/April 2001.

[6] M. E. Fagan. Advances in Software Inspections. *IEEE Transactions on Software Engineering*, 12(7), July 1986.

[7] James D. Herbsleb and Rebecca E. Grinter. Slitting the Organization and Integrating the Code. In *Proceedings of the* 21^{st} *International Confernce on Software Engineering*, pages 85–95, Los Angeles, California, USA, 1999.

[8] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An Empirical Study of Global Software Development: Distance and Seed. In *Proceedings of the* 23^{rd} *International Confernce on Software Engineering*, pages 81–90, Toronto, Canada, May 2001.

[9] Fogel Karl. *Open Source Development with CVS.* CoriolisOpen Press, 1999.

[10] Werner Kobitzsch, Dieter Rombach, and Raimund L. Feldmann. Outsourcing in India. *IEEE Software*, 18(2):78–86, March/April 2001.

[11] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* The University of Chicago Press, Chicago, USA, 1970.

[12] Glyn Moody. *Rebel Code: Inside Linux and the Open Source Revolution.* Perseus Publishing, New York, NY, 2001.

[13] D. Parnas. On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, December 1972.

[14] E. Raymond. The Cathedral and the Bazaar. See `http://www.tuxedo.org/~esr/writings/cathedral-bazaar`.