

An Innovative Approach to Managing Distributed Teams

Anil Hashia

hashiaa@seattleu.edu

John Whelan

whelanj@seattleu.edu

Ruchi Shewaramani

shewaram@seattleu.edu

Seattle University

Department of Computer Science & Software Engineering

901 12th Avenue, P.O. Box 222000

Seattle, WA 98122-1090

(206) 296 5510

Abstract

The purpose of this paper is to identify key practices used by open source¹ and commercial software² developers to improve the performance of distributed teams. To meet this objective, we identified major issues affecting distributed teams. After face-to-face and structured telephone interviews with team managers of commercial software companies we established a list of common problems and concerns. We also received input from open source experts regarding the distributed nature of their development. Using surveys, interviews and questionnaires, we compiled a collection of best practices. These best practices should correct common mistakes and typical problems that hinder productivity of distributed teams.

Keywords: Distributed Teams, Remote Teams, Global Software Development (GSD), Open Source Software Development (OSSD), Outsourcing

¹ "Open source" is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in [34].

² "Commercial software" is the model where the software developed by a commercial entity is typically licensed for a fee to a customer (either directly or through channels) in object, binary or executable code. The commercial entity often provides support, training, updates and other similar services needed by customers to efficiently use that software [33].

Table of Contents

1. Introduction	3
2. Understanding Problems with Distributed Teams	4
2.1 Communication Issues	5
2.2 Project Management Issues	5
2.3 Schedule Issues	6
3. Analyzing an Approach.....	6
3.1 Open Source Software Development.....	7
3.1.1 Communication	7
3.1.2 Management	8
3.1.3 Schedules	9
3.2 Commercial Software Development	10
3.2.1 Interview 1: Microsoft (MS)	10
3.2.2 Interview 2: RealNetworks.....	11
3.2.3 Interview 3: Boeing	12
3.2.4 Compilation of tools and practices.....	14
4. An Innovative Approach to Managing Distributed Teams	15
4.1 Data Analysis	15
4.1.1 Communication	15
4.1.2 Management	16
4.1.3 Schedules	18
4.2 Best Practices	19
4.2.1 List of 13 best practices for Distributed Teams	19
5. Conclusions	20
6. Acknowledgements.....	20
7. References	20
8. Appendix.....	22

1. Introduction

In this paper, we define a distributed team as "a group of people working on the same project, who are not [all] centrally located". Rather than everyone working from a central location, like an office building or an office complex, members of a distributed team work from wherever it makes the best sense; either for locality or economic reasons. Lately, this trend has grown to include *outsourcing*.

Software companies are increasing the use of distributed teams for their projects. Around 40 percent of the Fortune 500 companies use Global Software Development (GSD) [1]. Enterprises are under constant demand to employ remote developers to cut costs [3]. Presently, 61 percent of enterprises have some remote development and 50 percent outsource some or all of their software development [5]. Distributed teams offer advantages in terms of skills and cost but at the same time, there are challenges involved with communication, management and schedules.

We studied GSD at Microsoft, Real Networks, Boeing and other companies by interviewing project managers from these companies. We investigated the issues they face and how they are resolved. At the same time, we also studied open source software development (OSSD) which focuses on collaboration. OSSD relies on collaboration of developers from around the world who rarely meet face-to-face, and coordinate their activity by means of email, IRC and mailing lists. OSSD has produced successful projects like Linux and Apache. SourceForge.net hosts over 70,000 different Open Source projects and has over 700,000 registered users [2] that give an indication of the popularity of the open source model. We also conducted an online open source survey. The idea of the survey was to get a better understanding open source model and how it contributes to the effective collaboration of remote developers.

In Section 2, we describe different problems distributed teams experience in areas of communication, management and scheduling. In Section 3, we then introduce an approach to handling these problems. The approach derives from the open source community and commercial software development interviews. In Section 4, we discuss our approach for a successful distributed team based on the results we have gathered. Using examples and statistics, we describe best practices for overcoming common obstacles with distributed team's productivity. Figure 1 in next page shows our timeline and various steps followed during that timeline.

MSE Project

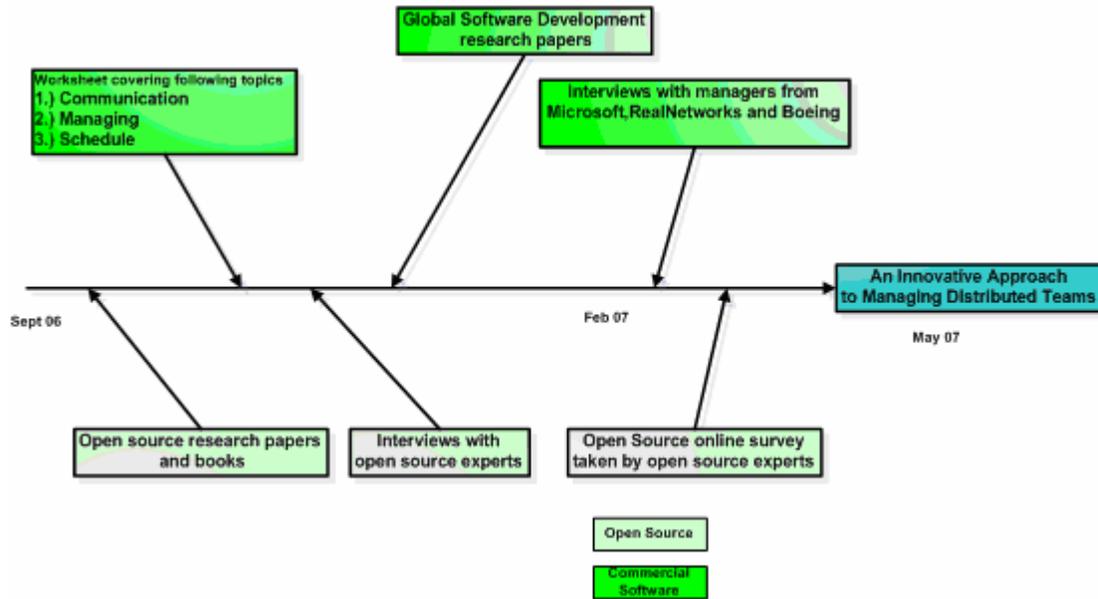


Figure 1: Our Research approach involved studying distributed teams from both open source and commercial software models. The rectangles above the timeline represent work done from commercial software perspective and the rectangles below the timeline represent work done from open source perspective.

2. Understanding Problems with Distributed Teams

In distributed teams, face-to-face communication between sub-teams rarely happens and collaboration is mainly using application of collaborative technologies (e.g. phone and Internet). During our research, we discovered that the key motivations for creating dispersed development teams are to leverage local expertise, reduce time-to-market and reduce development costs [3]. On the other hand, there is a lot of overhead associated with managing distributed teams in terms of intra-group communication, cultural differences and program management. If not properly managed, scope creep, poor quality work, as well as budget and schedule overruns can occur [3].

Some of the challenges of managing distributed teams relate to conflict and trust. Conflicts can arise from the use of processes and tools, differences in work ethics and cultural perspectives. Lack of trust is due to miscommunication, inability to do on the spot verification and lack of visible progress. In some cases, quality of work can be difficult to enforce as well. While managing distributed teams, it is important for the project manager to reduce conflict and build trust.

2.1 Communication Issues

The primary reason for communication issues with distributed teams is the lack of physical proximity and work-cycle synchronicity (i.e. members working in different time zones) [3] [4]. Since there are no face-to-face meetings, remote teams easily get out of sync. Most of the project managers we surveyed are of the opinion that effective team communication is of paramount importance for the success of the project [3]

Establishing and maintaining mutual knowledge, the knowledge communicating parties share, is central to the success of a distributed team. C.Cramton identifies five types of problems that result in failure to maintain mutual knowledge [15]. It is important to identify and correct these problems in distributed teams. Table 1 lists above mentioned problems.

Problems in remote teams resulting in failure to maintain mutual knowledge

1. Failure to communicate and retain contextual information
2. Unevenly distributed information
3. Difficulty communicating and understanding the salience of information
4. Differences in speed of access to information
5. Difficulty interpreting the meaning of silence

Table 1: Problem in maintaining mutual knowledge by C.Cramton

2.2 Project Management Issues

The complex, usually uncertain, and highly interdependent nature of project tasks, together with the geographical, temporal, structural, and cultural gaps characteristic of distributed teams, make the management of projects a relatively complex undertaking [7]. In general, project managers have difficulty in managing remote teams compared to managing centrally located teams [3]. Project Managers lack direct visibility of a remote team's progress, making managing risks difficult. As a result, distributed teams require additional project management and planning [3].

Globally distributed teams are comprised of people from different national cultures who have their own set of work ethics, beliefs and values. An individual attitude toward peers and respect for "authority" varies in different cultures. These cultural differences can lead to surprises regarding commitments, expectations and milestones. In some cultures, there is a tendency to say "yes" to everything [3]; a person dutifully executes the work delegated even though they may doubt their ability to accomplish the task. By contrast, in other cultures, developers will not commit to anything they cannot accomplish. They would be upfront about their limitations and offer their own suggestions. Particular societies tend to have distinct ways of working which can prove problematic when attempting cross-border collaboration. For example, Indian software companies have found they need to approach communication with U.S. and Japanese clients in very different ways. U.S. clients normally work with extensive written agreements and explicit documentation, reinforced with frequent and informal telephone and email contact. By contrast, Japanese clients tend to prefer verbal communication, tacit and continuously negotiated agreements, and less frequent but formal use of electronic media [16].

A paper by Ravi and Raymond [17] and other research papers quantify cultural differences in terms of Power Distance Index (PDI), Individualism (IDV) and Uncertainty Avoidance Index (UAI). The characteristics of these indexes are

- Small PDI means that subordinates participate more in decision-making activities
- High IDV means that everybody has the right of his own opinion
- High UAI means that everything needs to be carefully planned

It is important for a project manager to understand and manage these cross-cultural communication challenges. Table 2 shows PDI, IDV and UAI of US, India, China, Japan, Belgium and Ireland [32].

Power Distance Index (PDI): Individual's acceptance of authority
Individualism (IDV): Preference of people to belong to a loosely versus a tightly knit social framework
Uncertainty Avoidance Index (UAI): Individual's unwillingness to accept risk

	US	India	China	Japan	Belgium	Ireland
PDI	40	77	80	54	65	28
IDV	98	48	20	46	75	70
UAI	46	40	40	92	94	35

Table 2: Cross Cultural Communication metrics [32]

2.3 Schedule Issues

An essential element of project management is to meet schedules. It is also a requirement of a project's stakeholders. A project's schedule clearly defines when work is to be completed. Since there is little control over distributed teams because of geographical distance, it makes managing and controlling schedules difficult for a project manager.

3. Analyzing an Approach

The environment in which distributed teams operate is inherently difficult. Problems described in Section 2 can cause a project to stall and exceed its budget [35]. Because of their inherent nature, these problems can be difficult to spot and take longer to surface. Fortunately, the open source community matured in this environment and tackled these problems early on. Not only did they address the obvious problems previously mentioned, but some subtle ones as well [34].

3.1 Open Source Software Development

Distributed by nature, open source software development encourages developers to contribute regardless of their location. The community even goes beyond this by sharing collective decisions during development and making them available to everyone in the public domain. In this sense, not only are their members distributed, but their roles are as well. In addition to being distributed, the open source community as a whole has also been somewhat successful [2]. This would not be unusual, except for the extraordinary conditions under which this model operates. Many of these "conditions" include attributes that would spell out complete failure in the commercial world. However, since some open source projects succeed, we can conclude they must be doing something right.

In fact, the success of any open source project can be considered quite amazing. After all, a non-paid work force that works whenever they want provides much of the effort. There are no unions, bonuses or reasonable incentives to ensure that employees complete the work in a timely manner. Schedules and management are often relaxed as well³. So, how do they succeed? The truth is - many do not. Just starting an open source project is a challenge. Most start either by individuals who want to solve a particular problem or by corporations that decide to take an existing project and "open" it up. In either case, there are many obstacles to overcome with little guarantee of success as a number of projects never make it to their first release [12]. However, the ones that are successful, Linux, Apache, etc. have been so for a couple of reasons. These include:

- They were well-planned and orchestrated projects needed by a collection/group of people.
- They were lucky.
- They followed common guidelines established early on to help foster their projects and community.
- Other aspects, such as the Internet, standardization, improved legalities, etc. helped as well.

Of all these, the guidelines they followed differentiated them the most from commercial projects. These guidelines or principals are unique to the open source community. They help promote an open environment, one that ensures high quality and unmitigated participation. We believe that companies can apply these principles toward solving the problems of distributed teams.

³ *Not all open source projects consist only of non-paid workers. Corporate funding of free software development is quite common. Much source development is informally subsidized or in a company's best interest [9]. Sources estimate that 40% open source participants are paid [19].*

3.1.1 Communication

Communication takes place in open source projects the same way as it would in any other type of project, except it is not verbal. This means no private discussions or teleconferences. All communication must be in written form, so that teams can archive and retrieve them when needed. This guarantees a record of every discussion. Communication tools like IRC, email and instant messaging are common, as well as general use of mailing lists, forums and wikis. As long as someone can archive the type of communication, it is encouraged. The importance of archiving communication is

significant as it allows issues that resurface, to be resolved quickly without reworking them. This saves time and keeps everyone on the same page, despite his or her experience. In essence, it creates a "shared depository of project wisdom" [9].

Another benefit of nonverbal communication is the ability to interpret it, since written words are verifiable and not subject to the same misinterpretations as if spoken. With verbal communications, people located in other countries may have accents that are difficult to understand. These accents do not necessarily appear on paper. However, there is a trade-off. Tones and body language do not translate on paper either. Depending on the nature of the discussion, certain elements could be lost if not presented verbally. In addition, the use of humor or lack thereof, can be another advantage. Humor may give the wrong impression if spoken, but fortunately is seldom used in writings.

The open source community is also a large supporter of public discussion. All conversations are completely open and take place in a public forum. Anyone can participate in these discussions, regardless of their tenure with the project. Since contributors are usually at a distance from one another, open source, like any distributed team, has the disadvantage of relatively weak coordination, which impedes the ability to collaborate. However, this disadvantage may actually be a hidden strength. Research suggests that the organizational structure of a software project directly manifests itself in the design of the product. In particular, they believed that the tight organization of a proprietary project would result in a tightly integrated program with many inter-dependencies, while the loose organization of an open-source project would lead to a modular design with fewer inter-dependencies [18]. This concept is apparent in such open source projects as Linux. Linux is comprised of subsystems sharing few inter-dependencies among them. This loose organization makes Linux highly adaptable and very effective.

3.1.2 Management

Open source projects to some degree are self-managed. Most contributors take it upon themselves to manage their own work. When conflicts occur, they are usually resolved in one of two ways, depending on which model the project follows: dictatorship or democratic.

In a dictatorship model, a senior member who is well respected and admired by the team settles conflicts. This would be someone like Linus Torvalds, whose decisions to this day greatly influence the Linux open source project. The Wikipedia community acknowledges founder Jimmy Wales as that project's benevolent dictator. Like Linus Torvalds, he reserves the right to make unilateral decisions, even though in practice, he rarely exercises this right [11]. In the dictatorship model, if someone does not agree with the decision of the dictator, that person has the right to "fork" the project. This means that person would split from the group and go their own way, starting another open source project with their approach ⁴.

⁴ *Note: The ability to "fork" would not be possible in a commercial environment. However, this would give members a great opportunity to propose a compromise that could "fork" from the original direction.*

In a democratic model, participants settle conflicts by voting. All votes are equal; however, the community does not allow all members to vote. Usually, this right is only reserved for active members who are currently working on the project. This eliminates any party not directly involved with the project from altering the outcome of the vote.

In general, there are no official positions in an open source project. There are no titles, labels or the concept of roles. No one assigns positions; participants choose them. Everyone works on whatever part of the project he or she is particularly interested in or skilled at ⁵. Participants gain status from involvement and participation, not from credentials or years of service.

There is a popular misconception that open source projects are run by individuals void of any personality or "people skills". In reality, the opposite is true. People skills; such as patience, understanding and kindness are necessary ingredients to the success of any open source project [3]. These skills are usually in abundance in an open environment, where there are no favorites and everyone has the same sense of equality. Many projects succeed because the people within them have the ability to work with and help each other.

3.1.3 Schedules

In an open source project, individual schedules are flexible. Typically, deadlines are enforced by personal motivation more than anything else. Most contributors work when they want or when time is available, since many consider their open source work as a part time job. Time spent comes from dedication, more than obligation. So, what drives this dedication?

Most members of an open source project are not in it for the money. Instead, they are chasing a dream. They are motivated by a variety of reasons and are commonly underutilized in their current, often corporate, roles [12]. Solving interesting problems, learning opportunities, correcting irritating bugs, and even self-promotion drive them. While peer recognition is not unique to the open source community, it is a strong motivating factor. Unlike the typical work environment, open source peers share a common ambition that is far greater than any monetary goal they could ever achieve at work. They share a belief they can actually change the world or take on a corporate giant. They believe their products benefit the greater good, not just the bottom line of a corporation [36]. As a result, recognition from this type of peer is indispensable. Another motivating factor is the means by which participants can improve their programming skills through active peer reviews. Software code contributions are typically subject to intense peer reviews both before and after a submission become part of the official code base [19]. All errors and improvements found are communicated back to the original author, as well as being archived. This forum provides excellent feedback and allows developers the opportunity to improve their skills without having to suffer professionally.

⁵ *Primarily, workers volunteer for assignments. Rarely does the community assign a task to someone, unless the task relates to a particular bug that corresponds to the work they did.*

Open source projects do not usually have static release dates, but instead rely on frequent feedback from their users to ensure their survival. Common wisdom says, "Release early, Release often" [20]. Most projects do not release alphas, betas, or even prototypes. Instead, they release stable, high-quality software ⁶ capable of immediate use. One technique the open source community applies when building its schedules, is the use of "stories", made popular by XP (eXtreme Programming). They allow the customer to describe functionality by communicating them in terms of a story. A story is just a brief description of a feature from a customer's point of view. Stories have to be short, concrete and testable. The open source community found this technique worked nicely when applied to schedules. By building schedules around stories, it is possible to customize a project's work force for each release. Every story scheduled for a release is tested. If all tests are successful, then the story is finished and released.

⁶ *One goal of open source development is to produce better quality, higher reliability and more flexibility software [34]. Of course, not all projects reach this goal.*

3.2 Commercial Software Development

Open source is not the only environment to have success with distributed teams. Their use is becoming a common practice in commercial software, as well. We took a closer look at three companies; Microsoft, RealNetworks and Boeing, which all employ distributed teams. The following interviews describe the issues these companies faced, and action they take in trying to resolve those using changes in communication, management and schedules.

3.2.1 Interview 1: Microsoft (MS)

Microsoft is headquartered in Redmond, WA, USA. As of June 30, 2006, Microsoft employed approximately 71,000 people on a full-time basis, 44,000 in the United States and **27,000 internationally** [29].

We had a discussion with a Research Manager at Microsoft who studied collaboration processes followed by different Microsoft distributed teams located in various parts of the world. His goal was to improve the collaboration process between remote teams. His study involved remote teams listed in Table 3.

Country	Location	Nature of work	Size (Approx.)
Ireland	Dublin	Development	1200
India	Hyderabad	Development	1500
China	Beijing	Research	100-200
China	Shangai	Development	100-200

Table 3: Remote Teams involved in the study

3.2.1.1 Communication

In this interview, we learned that different teams in Microsoft have different needs for communication and handle them in ways that best suit each teams' requirements. For example, the Dynamics team in Denmark works on a large project that requires less management from the Redmond head quarters. On the other hand, the Windows Vista team needs day-to-day management. For day-to-day communication, they largely use email. However, using email across time shifts does not work well for making quick

decisions. It calls for more autonomy on the part of the distributed team, as it is difficult to reach a conclusion over email given time differences. Teams also use conference calls and desktop sharing to collaborate with each other. For example, the Office Online team has live meetings with their globally distributed team every week.

3.2.1.2 Management

To mitigate problems with cultural differences, Microsoft encourages the exchange of people between teams in other countries to form personal relationships. They have found the ideal exchange time to be six weeks. This gives people a sense of belonging and understanding with the remote team.

There is much emphasis on developing the team morale. Microsoft refers to geographically distributed software development as Global development instead of remote development.

There is a mixture of contractors and employees in all development centers. Microsoft is committed to all the development centers and employees. Microsoft hire the employees with the expectation that they will pursue long term careers with Microsoft, as is the case with Redmond based full time hires. For managing outsourced clients, Microsoft follows the "peer contracting" model. Microsoft signs a contract with the client that explicitly covers any assumptions made about when deliverables are due and how they need to be developed. It also covers how problems will be escalated to the Microsoft team back in U.S if they arise.

3.2.1.3 Schedules

Management of project schedules uses a combination of technology. They utilize SharePoint, MS Project, MS Excel and Visual Studio Team System across different teams. In our discussion, we learned that project management is more important during planning and development phases. Regular interim deliverables keep the global teams on track with the overall schedule.

3.2.2 Interview 2: RealNetworks

RealNetworks provides software and services related to Internet media. It is in Seattle, WA, USA. As of December 31, 2006, RealNetworks had 1,594 full-time employees and 55 part-time employees, of which 1,013 were based in the Americas, **459** were based in **Asia**, and **177** were based in **Europe** [30]. The Team manager that we had discussion with is managing Offshore Development Center (ODC) in Pune, India for some of the development, testing and maintenance in his project.

Some of observations we made from this discussion are:

- ODC teams are well suited for quality assurance and testing.
- Corporate headquarters in Seattle handles program management.
- Training is necessary for understanding cultural issues.
- Contract should specify penalties for not completing work in time.
- Exchange of teams for 1-2 weeks is good way for building trust.
- Cost is less (around one-fourth in one project), schedule is longer and quality is the same or slightly worse.

3.2.2.1 Communication

RealNetworks uses e-mail for exchanging information, communicating status changes, delivering updates and asking/answering questions. They use conference calls twice per week at minimum to get status and updates. They use Instant Messenger for near-instantaneous transmission of messages to people over a network, which also provides an indication of their presence. English is the common language used and it is a fundamental requirement for vendors who provide off shore development services.

3.2.2.2 Management

Both ODC and local teams have a similar team structure. Leads are responsible for their discipline (Dev/QA) that rolls up to project manager. The idea behind having a similar setup for ODC is to have consistency and to leverage existing processes. Some individual contributors in ODC work closely with leads in Seattle. Overall, the project manager in Seattle controls the project. Recruiting for distributed teams is always a challenge. It needs comprehension of culture and knowledge of business environment in which ODC operates. RealNetworks handles cultural issues by interviewing people that have worked with the culture and by providing classes on International business. A program Manager's responsibility is to ensure that these people are flexible with offshore teams and mature enough to work collaboratively with other cultures. Since distance is significant, teams need more supervision and management. Setting expectations, training and close interaction is necessary. They discourage an "Us versus Them" mentality by treating both locations as a single team with a single focus and separate deliverables. They celebrate success jointly and distribute items like T-shirts and coffee mugs to all teams. Senior Management periodically visits remote sites and people from remote sites visit the Seattle location for face-to-face interaction. These visits help to build trust if RealNetworks needs the ODC on a long-term basis. They use Plone Content Management System [8] for managing information (specs, plan of records, design, schedules, project review notes etc).

3.2.2.3 Schedules

Processes used for scheduling are stand-up meetings, lead meetings, etc. Initial planning is at Seattle. ODC is involved at a latter phase of the software development life cycle. Weekly project reviews are for formal status of progress, issues and needs. The contract with ODC is a very important document that clearly sets deliverables, quality of work and milestones. RealNetworks holds the ODC accountable through terms and conditions of the contract.

3.2.3 Interview 3: Boeing

Boeing is a large company that operates in many countries. Originally started in Seattle, Boeing's workforce level as of December 31, 2006 was approximately 154,000, including **1,300** in **Canada** and **3,700** in **Australia** [31].

3.2.3.1 Communication

Starting in the late 1990s, Boeing expanded dramatically because of a merge with McDonnell Douglas. Before this merger, about 80% of Boeing's work force was located in the Puget Sound and most of its business was conducted locally [13]. This included attending meetings face-to-face. Following the merger, only 40% of the company's work force remained in the Seattle area [13]. This caused the cost/benefit ratio of personally attending meetings to change. It was no longer feasible to attend every meeting; instead, a virtual presence would have to do. Initially, Boeing made use of audio and video conferencing. In its early implementation, these technologies did not prove to be sufficient. Videoconferencing did not provide high enough resolution for engineers to present their highly detailed designs. In addition, the bandwidth required for such images was expensive and therefore not readily available at the time. A better alternative turned out to be data conferencing. Data conferencing allows users to share data among large groups simultaneously. Within Boeing, it has grown to include many different tools. These include WebEx, Net Manager and Oracle Web Conferencing, as well as others. Additional file sharing tools like Share-point and Windchill have also added value to Boeing's distributed environment.

3.2.3.2 Management

Due to their size, Boeing utilizes almost every type of distributed team imaginable. In addition, they handle each type of distributed team throughout the company differently. In many cases, distributed teams can include entire organizations. Nevertheless, some best practices are uniformly represented throughout the entire company. One such practice, mentioned above, is the constant use of technology. Another is the use of education. To address problems that might occur between cultures, Boeing utilizes "culture classes". Boeing teaches these classes to inform workers about differences among cultures in a distributed environment. They specialize in unique characteristics each culture has concerning work ethics, honesty, social issues, etc. [21] In addition to courses, Boeing has seminars on culture awareness provided by experts in each culture it works with. Boeing also supplies virtual employees and telecommuters with additional training as well. This ensures that no matter what type of distributed team a Boeing employee is involved with, help is available.

3.2.3.3 Schedules

To address the scheduling problems of distributed teams, Boeing employs team calendars. These calendars are either located online or within Microsoft Outlook. They automatically account for different time zones and give a warning when a meeting or significant event is about to occur [3].

Boeing uses a combination of technology for the management of project schedules. They use SharePoint, MS Project, MS Excel and Visual Studio Team System across different teams. In the discussion, they also indicated that project management is more important during the planning and development phases. They plan regular interim deliverables to keep the global teams on track with the overall schedule.

3.2.4 Compilation of tools and practices

Below is a list of the tools and practices we compiled in our research.

	Open Source	Microsoft	Real Networks	Boeing
Communication				
E-mail, Instant messenger	✓	✓	✓	✓
Tele conferencing	x	✓	✓	✓
Video conferencing	x	Not as much as tele-conferencing	Not as much as tele conferencing	Not as much as tele conferencing
Blogs for team communication	✓	Gaining popularity	-	x
IRC	✓	x	x	x
Mailing List	✓	Optional	Optional	Optional
Data conferencing	Not as much as commercial	✓	✓	✓
Management				
Culture Class	x	✓	x	✓
Send people onsite for team morale	x	✓	✓	-
Follow same team hierarchy	x	✓	✓	-
Self Directing Democratic model	✓	x	x	x
Schedule				
Team Calendars	-	✓	✓	✓
Regular milestones	-	✓	✓	✓
Contractual obligations for meeting schedule	x	✓	✓	✓

Table 4: List of tools and practices

✓-Yes

x- No

- Unknown

4. An Innovative Approach to Managing Distributed Teams

We divided this section into two parts. In the first part, Data Analysis, we present key findings from our research. We analyze data from surveys of open source and commercial distributed teams. In the second section, we use this analysis to create a list of best practices that you can use as a reference to guide distributed teams.

4.1 Data Analysis

4.1.1 Communication

Our analysis found that on a daily basis our respondents preferred written communication to verbal communication. Not surprisingly, mailing lists, discussion boards, forums and chat tools are the main components of communication in distributed environment, whether it is open source or commercial. Private discussions are not encouraged. Open source teams largely use Wikis, while their use is gaining popularity among the commercial teams. Distributed teams are handling most language and accent issues with written communication. However, from our interviews, we also discovered this could lead to rare instances where written communication produces long email threads between people. Therefore, it is important not to take written communications too far.

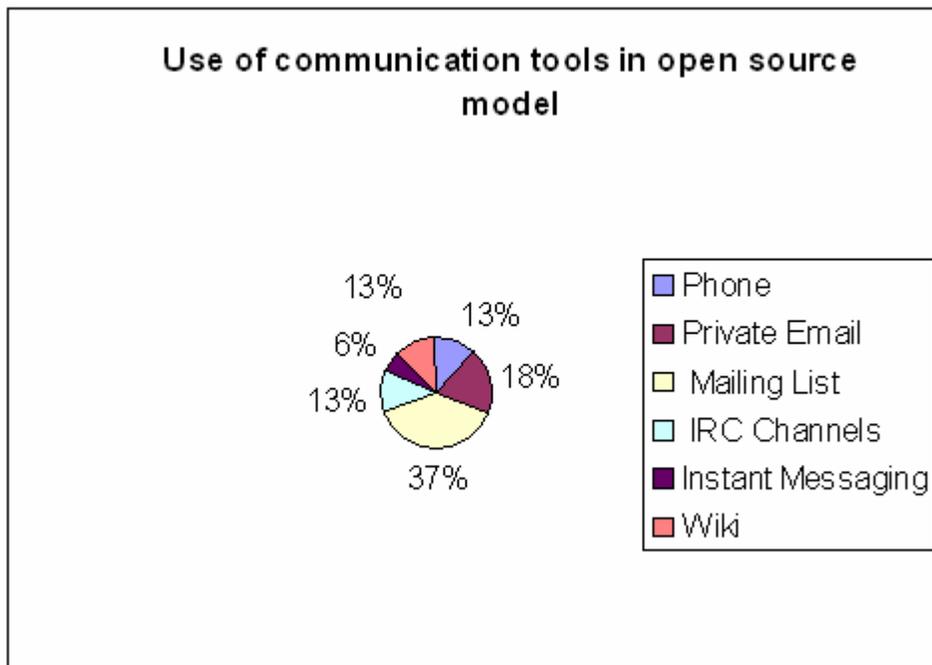


Figure 2: Communication tools used by the open source community

Figure 2, on the previous page, shows the percentage of communication tools used by open source distributed teams. Interestingly, only 13 percent of distributed teams surveyed said they use verbal communication. In contrast, all the managers of commercial projects stated that individuals prefer to work with each other in person. They dislike emails and prefer regular teleconferencing to achieve the same effect. Commercial software teams encourage regular interaction between distributed teams through virtual or onsite meetings to ensure a sense of belonging. In contrast, open source contributors have an inherent sense of belonging to the project as they are there because they want to and not because they have to. In the interest of improving communication, commercial projects need to focus on mitigating language issues. To do this, managers identify a point of contact that understands the different cultures. The role of this contact is to moderate and ensure that the two geographically distributed teams communicate well. We also learned that when managers recruit to form a distributed team, "people skills" and the ability to understand and respect different cultures is a desired quality.

4.1.2 Management

4.1.2.1 Process

The software industry places an emphasis on process. There are trade-offs between too much and too little process. Following a well-defined and documented process, that is clearly communicated and understood by the team, is critical to the success of that team. This is especially true if that team is working in different time zones and has different cultural backgrounds.

4.1.2.2 Conflict

Conflict is an integral part of cooperative work in many work settings [26]. Conflict resolution is a task in itself with a distributed team and is further complicated when entire teams are in disagreement. We surveyed the open source industry to find out how often conflicts arise. Contradictory to our expectations and in line with the success of open source projects, we learned that the percentage of conflicts occurrence is not very high.

However, intra-group conflict is inevitable. Managing conflict is more important than the conflict itself [27]. Figure 3, on the next page, shows the percentage of contributors who leave open source software projects due to conflicts. As seen in the graph, large number of OS contributors responded that rarely do people leave projects due to conflicts. This shows that the Open source projects are well managed and conflicts are well resolved.

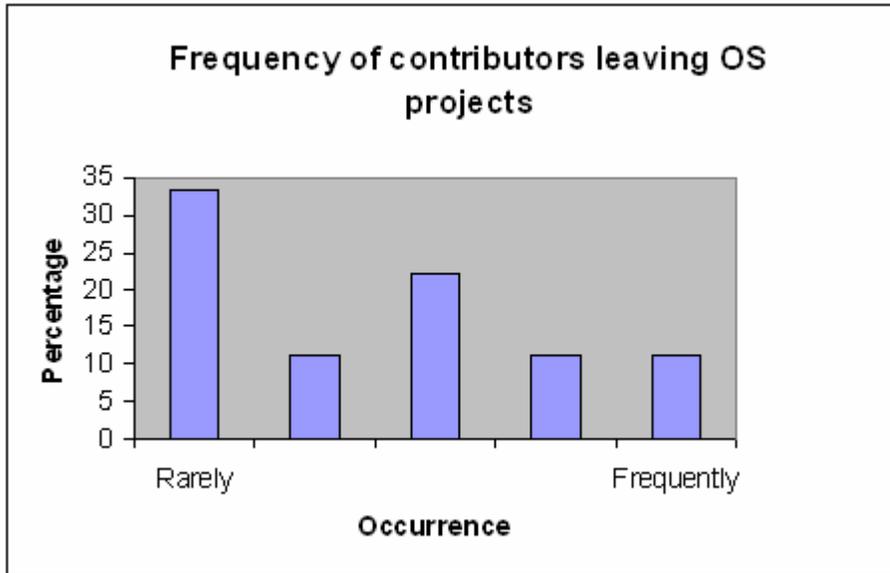


Figure 3: Frequency of contributors leaving OS projects due to conflicts

We were interested in learning how conflicts are resolved in the open source community with the absence of corporate structure and a higher authority to make final decisions. Figure 4 shows what percentage of teams follow different methods for conflict resolution in their teams. A large number of teams resolve conflicts by team discussion. Open source teams resolve only a small number of conflicts by taking technical expert's advice.

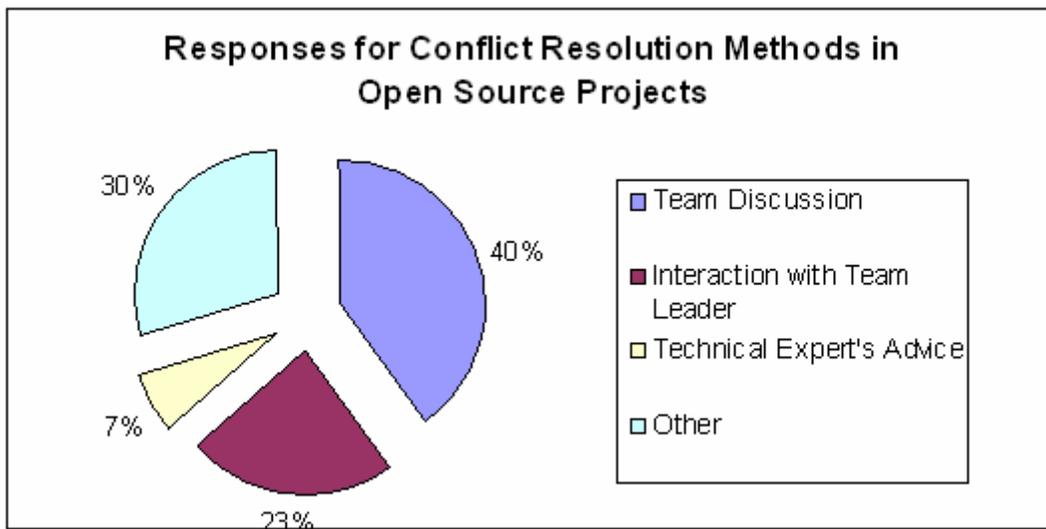


Figure 4: Responses for conflict resolution methods followed by OS projects

4.1.2.3 Team style

Stature in open source teams comes from direct participation and involvement. There is little emphasis put on tenure or years of experience, as most contributors do not disclose their background. Instead, the structure of a team defines itself. Natural leaders

ascend to the top and equally spread their contributions throughout the project. This also occurs in commercial environments, but is less likely to be a result of direct participation and involvement.

Cultural awareness: Though this is not widely practiced by the companies, almost everyone we interacted with during our research expressed a positive attitude towards implementing cultural awareness. We would suggest companies arrange cultural awareness sessions and familiarize their employees with the customs and traditions of the different cultures with which their company interacts.

Us Vs Them: Companies can achieve a sense of belonging and familiarity for their outsourced teams by sending people to the off shore centers. This works in two ways. Sending a senior member of the company to the off shore center boosts the morale of the off shore team and makes them feel that they are a part of the team and not just cheap labor available to do programming and maintenance work. Another helpful trend is to send programmers and testers who work in U.S centers to centers in India or China to work with their fellow testers and programmers with whom they interact on daily basis [3]. This helps tremendously in solving the "us vs. them" approach. Since, it is not financially viable to send the entire team of programmers and testers to the offshore centers, the more practical and implemented option is to send program managers and test managers who can ultimately be the moderators between the two culturally different sides.

4.1.3 Schedules

Both open source and commercial projects use a short milestone approach to keep the members of distributed team on track. Almost all open source members interviewed responded that communication among contributors is not an essential element for keeping projects on track. This is in contrast to most of the commercial projects where the communication between a contributor and manager is critical for the success of the project. When interviewed about the level of difficulty of tracking open source schedules, most indicated that it was moderately difficult. In addition, more than half responded that they push the schedule during testing and bug fixing phase. Keeping open source projects on track is more difficult compared to commercial, because turnover is much higher. The use of tools for tracking schedule is pervasive across all software projects, though the actual tools themselves may differ.

Approach to schedules: Schedules are supposed to be static timelines of a product's development lifecycle. Unfortunately, they are seldom static. Both commercial and open source projects are subject to schedule problems or delays. Both environments take delays seriously and incorporate methods to minimize schedule problems. Open source limits its releases and often shifts resources when the need arises. They also frequently employ the use of "stories" to help manage schedules and deadlines.

4.2 Best Practices

4.2.1 List of 13 best practices for Distributed Teams

Following is the list of best practices for distributed teams with a short description of each.

Best Practices	Description
Communication	
1 Communication should be in written form	Written communication helps avoid accent/language issues and can be archived for tracking purposes. Using tools like mail lists, IRC, IM and Net Meeting.
2 Create public forum for discussion	Brings diverse ideas and viewpoints to the project. Can be archived as a shared depository of project wisdom.
3 Avoid private discussions	Private discussions obstruct team consensus and hinders development by creating gaps in communication.
4 Take advantage of the latest technology available	Collaboration tools are constantly improving and evolving. Encourage use of portals, wikis and blogs.
Management	
5 Practice people exchange	Send team members onsite for 4-6 weeks. For larger co-located teams, this is a great way for building team morale and bridging cultural gaps.
6 Consensus based conflict resolution	This works better than having an expert make the final decision, unless it is necessary.
7 Promote a sense of equality	All team members should feel their input is important and equal to everyone else's. Treat all members alike, irrespective of their location. Discourage favoritism.
8 Appreciate and reward good work	Participation and involvement should determine status. Active participation is a better way to measure someone's abilities.
9 Participate in cross-culture training programs	These training programs provide a gateway to understand and respect different cultures.
10 Encourage people skills	Working and reasoning on a personal level allows people to interact more naturally.
Schedule	
11 Regularly spaced deliverables	This helps determine if a team is on track on a regular basis.
12 Release early, release often	Releasing early exposes bugs sooner and releasing often allows you to fix them quickly. Working software is a good measure of success.
13 Schedule all code reviews to be open	Code reviews should be open to everyone in the company.

5. Conclusions

Our research focused on understanding distributed teams, their salient characteristics, the communication issues they face, the technical issues involved, the cultural differences that arise and the managerial implications. We looked to the open source community and three global companies to provide us with some guidance. From each source, we acquired a deeper knowledge base and were able to compile a list of best practices. We agree there is still more work that needs to be done in this area. Distributed teams, while somewhat popular, are still experiencing problems that do not generally affect centrally located ones. We believe the open source community, along with findings from successful companies, can provide a model for the rest of us to follow.

6. Acknowledgements

The following people supported the research described in this project:

- Karl Fogel of Open Source
- Mike Kelly of Microsoft
- Robin Mckone of RealNetworks
- Steven Poltrock of Boeing

7. References

- [1] Global Business Tecnology, Nassacom, 2000
- [2] http://sourceforge.net/donate/index.php?group_id=23067
- [3] Personal interviews with Program Managers from Microsoft, RealNetworks, Boeing and open source experts.
- [4] http://www-users.cs.york.ac.uk/~kimble/teaching/mis/Distributed_Team_Work.html
- [5] Evans Data, "Enterprise Development Management Issues," December 2001
- [6] Culture Surprises in Remote Software Development Teams, Judith S. Olson, Gary M. Olson
- [7] Virtual teams: an exploratory study of key challenges and strategies Guy Paré, Line Dubé
- [8] <http://plone.org/>
- [9] Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel @2005
- [10] www.boeing.com /employment/ Copyright @ 1995-2007
- [11] Analysis of Open Source principles in diverse collaborative communities. Jill Coffin Copyright @ 2006
- [12] Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism) Nikolai Bezroukov Copyright @ 1999

MSE Project

- [13] Expanding the Horizons of Requirements Engineering: Examining Requirements during Groupware Tool Diffusion Gloria Mark, Mark Bergman, Steven Poltrock Copyright 2004
- [14] Preliminary Insights into the In-Group/Out-Group Effecting Partially Distributed Teams. Haiyan Huang, Rosalie Ocker
- [15] The Mutual Knowledge Problem And Its Consequences For Dispersed Collaboration Catherine Durnell Cramton Organization Science, 12 (3), 346-371, 2001
- [16] Managing Cross-cultural Issues By S. Krishna, Sundeep Sahay, and Geoff Walsham. Exploring research-derived best practices for effective management of global software teams in global software outsourcing
- [17] Effects of Culture on Control Mechanisms in Offshore Outsourced IT Projects. Ravi Narayanaswamy and Raymond M. Henry
- [18] Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. Alan MacCormack, John Rusnak, Carliss Baldwin Copyright @ 2005
- [19] Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Project. Karim R. Lakhani, Robert G Wolf. Copyright @ 2005
- [20] Five Lessons Open Source Developers Should Learn from Extreme Programming {pseudonym } chromatic :: author of Extreme Programming Pocket Guide [chromatic promotes free and open source software for O'Reilly's Open Technology Exchange] Copyright @ 2003
- [21] <http://inside.boeing.com> – Internal website for employees; offering services and information.
- [22] Accelerating Software Development Through Collaboration. Larry Augustin, Dan Bressler, Guy Smith.
- [23] The process of joining in Global Distributed Software Teams. Israel Herraiz , Gregorio Robles, Juan José Amor , Teófilo Romera Jesús M. González Barahona.
- [24] SourceForge.net, Open Source hosting site - had 79,225 project registered as of mid-2004.
- [25] A case study of open source tools and practices in a commercial setting. Vijay K. Gurbani, Anita Garvert, James D. Herbsleb
- [26] Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration. Margaret S. Elliott, Walt Scacchi
- [27] Effect of Intra-Group Conflict on Packaged Software Development Team Performance. Steve Sawyer
- [28] Microsoft Office in Dublin, Ireland.
<http://www.microsoft.com/ireland/about/microsoftinireland.msp>
- [29] Microsoft's Annual Report 2006
- [30] RealNetworks Form 10K Filing Date: 3/1/2007
- [31] BOEING CO (BA) Form: 10-K Filing Date: 2/16/2007
- [32] <http://www.clearlycultural.com/geert-hofstede-cultural-dimensions/power-distance-index/>
- [33] <http://www.bsa.org/asia-eng/policy/upload/OS-Commercial-Analysis.pdf>
- [34] <http://www.opensource.org/>
- [35] The Social Structure of Free and Open Source Software Development - Kevin Crowston & James Howison Copyright 2005
- [36] Ten things you didn't know about open source - James Archibald Copyright 2007

8. Appendix

Attached are the documents from our research. These include the following:

- Commercial Software Development Worksheet
- Open Source Survey