The One-Minute Commute

Zack T. Grossbart

Share

Attribute

Remix

Share Alike

For Mary

Who read every page of this book
at least a dozen times.

About The Author

Zack has been working with and coaching remote teams at organizations like JP Morgan, 3M, Nortel, Hewlett Packard, and the United States Navy since 2001. He has served as a consultant to numerous Fortune 500 companies and is a consulting engineer for the Novell Compliance Management Platform. Zack began loading DOS from a floppy disk when he was five years old. He began working professionally with computers when he was 15 and started his first software company when he was 16. He has also been an IT administrator and a member of an advertising firm. Zack lives in Cambridge, Massachusetts, about a mile from Harvard University.

The One Minute Commute is Zack's first book and hopefully not his last.

About The Blog

Get more information about *The One Minute Commute*, find teleworking cheat sheets, and keep up with the latest from Zack at:

http://www.zackgrossbart.com

If you like this book you're karmically compelled to leave a comment on Zack's blog.

Photo credit: David Shopper

¶ Pilcrow Press
Cambridge, Massachusetts

Grossbart, Zack.
The One Minute Commute

This book is printed on paper-free paper

Praise for The One Minute Commute

CodeSourcery has always —from day one— been a high-tech company in which every single Sourcerer worked from home. In *The One Minute Commute* Zack Grossbart has captured the essence of how we function effectively. We'll be recommending this book to all future employees; *The One Minute Commute* is the how-to guide we never got a chance to write.

–Mark Mitchell, CEO of CodeSourcery

After learning the hard way, through a decade of home working, it is great to see much of the associated wisdom piled into Zack's single, pithy tome. With this on the shelf, when you next hit a sticky patch, the affirmation of sanity it can provide should be invaluable. You could even read it in advance to make life easier.

-Michael Meeks, OpenOffice, and GNOME
Distinguished Engineer, Novell, Inc.

Table of Contents

Preface: Doyle’s Room

Doyle Brunson played poker for the first time in 1951. After an injury in his junior year of high school ended his dreams of playing professional basketball, he played poker to pay his bills. He became a rounder, someone who made his living travelling and playing in different amateur poker games.

When Doyle got tired of looking down the barrel of a shotgun after beating someone who didn't want to lose, he moved to Las Vegas and worked to establish poker as a popular casino game. He became widely known as one of the strongest players in the world after he won $230,000 in the first ever $10,000 No Limit Texas Hold'em Poker Championship in 1976.

Doyle's skills in poker come from his ability to read a person's tells: the small changes in a person's expressions, how they smoke a cigarette, the way they hunch their shoulders, and hundreds of other subtle signals that tell him what cards his opponents are holding. He will stop and stare at his opponent, sizing them up, to figure out how to bet.

When other players started talking to Doyle about online poker in the early 1990's he dismissed them. Poker is all about reading other people and Doyle was certain online poker would never be real poker. He stayed in the casinos for years while other poker professionals established Internet poker rooms and online poker grew into a multi-billion dollar industry.

In 2004 Doyle changed his mind and became a telecommuter. He started Doyle's Room, an online service supporting all popular poker variants. He plays there often. So what changed between the early 1990's and 2004? Did the online poker companies offer him too much money to ignore? Doyle admits that the money was part of it, but that isn't the whole story.

What changed was Doyle. He started playing online and learned to read a new set of tells. He focused on how his opponents bet or how long they waited before playing. Doyle found an enormous amount of information while playing online and has been just as successful in virtual poker as he was in the real world.

Doyle said it best: "Online or in the real world, poker is poker. It's all poker."

Doyle overcame many of the hurdles you'll face working remotely and became a poker playing telecommuter. It cost him a lot of time and money to overcome those hurdles and there were many lessons he learned the hard way.

The lessons Doyle learned about how to interact with people in the virtual world are what this book is about. Now he knows all the tricks, sees the signs, understands what people are thinking, and reads the room even when he isn't in it. Being a successful team member means understanding what is going on for the rest of your team. Doing that remotely requires you to find alternative ways of communicating.

This book is about communication. Communication is the cornerstone to every software development team. It is the difference between being part of your team or just a screen name. Good communication leads to trust, productivity, and teamwork. Communicating well is the difference between success and failure.

Introduction

This is the book I wish I had seven years ago.

When I first started working as a remote engineer, I didn't know any of what you'll be reading in this book. I thought my success would be determined by how much code I produced and how good it was. I was completely wrong, and I had to learn most of the lessons in this book by making the mistakes. I decided to write this book in the hopes of sharing what I, and others, have learned through trial and error. It's the book I wish I'd read before I became a remote engineer.

On the first team I worked with as a full-time developer the build process took over an hour. A little while after I started I rewrote it and the build time went down to 10 minutes. It was the first piece of code I wrote that really made a difference to someone else. Engineers I had never met were thanking me in the hallway. I had been worried I wouldn't make it as a real engineer and this was a validation I needed.

The build process I wrote fell into disuse less than a year later when John, a coworker of mine who I counted as a friend, replaced it with his own build system. Having my code replaced didn't hurt nearly as much as the way my former friend did it: without any warning. I updated one day and it was just there.

To make matters worse, he had discussed it with almost everyone but me. I felt left out and excluded. John was my friend and he betrayed me. Not only were my feelings hurt, but my confidence was shot. I had lost a big step forward in my career, and I didn't know why.

I left one detail out of this story: between the time I wrote the build and the time John replaced it I started working remotely. John didn't ignore me out of malice or spite. He wasn't jealous of my talents or the success I had on the team; although I was secretly certain he was. He just didn't know how to talk to me now that I was permanently out of the office. Sure he could have picked up the phone, but I didn't make it easy.

When I started working remotely I almost disappeared. I boycotted IM and avoided conference calls. I never sent new emails and only replied to the ones I received when I was sure there was no other way. I wrote a lot of code and thought that should be enough.

In essence, I put out a big sign for my team saying "Leave Me Alone!" And John did. The saddest part is that it all could have been avoided by a few emails and conference calls.

Many people who start working remotely fail. They become isolated from their team and create a working environment with very little teamwork. The code the team produces gets worse and projects slow down or fail altogether. Many remote engineers falter because they lack an understanding of basic human communication.

Your team won't see you at your desk when they show up at the office, neither will your manager. You may have been working for hours, they just don't know. One of your biggest challenges is making sure that "out of sight" doesn't mean "out of mind."

Working remotely means accepting substitutes for standard communication. Instant messages replace a friendly chat and group meetings are now conference calls. Using these new, and in some cases old, alternative forms of communication effectively requires a fundamental understanding of how humans interact.

I have faith in teams. I believe that many people working together can achieve more than they can alone, but only if they work together well. Communication technologies will be cheaper, more widely available, and better in the future. Companies will take advantage of these technologies to build teams from around the world. The success or failure of those teams depends on the abilities of individual contributors to communicate well and stay close with their team, even when they are far away.

You Are Not Your Code

Reading this book will make you a better engineer. It has no code samples, algorithms, or discussions of any specific programming languages. Instead it focuses on communication, which will make you a better member of your team wherever you work.

It doesn't matter how good your code is if nobody sees it. You will never get promoted, rewarded, or even noticed if you can't talk and write about what you do. And that is really hard to do.

Coding is sometimes difficult, but the process is always simple. Write better code and your project will run better, have the features you need, and crash less. Write bad code and your project will not work very well. Communication and human interaction are not that simple. Some of the time it looks like everything is good when it isn't.

This book will help you communicate better because good communication is the most common casualty of distributed teams. It is also fundamental to every engineer. Software is about communication more than code. Code is the final result, but your communication with your team, your manager, and your customers is what gets you there.

What It Takes To Work Remotely

This book is for every type of remote engineer. Developers, testers, documentation writers, and anyone else involved in the software development process who works in a different location from the rest of their team. This includes engineers who work half-time from home and those who live on the other side of the world.

This section was strongly influenced by my conversations with Karl Fogel. Karl is a founding member of the Subversion project and the author of Producing Open Source Software: How to Run a Successful Free Software Project. I didn't speak with Karl for very long, but his influence on this project was profound.

Although working remotely can have a lot of benefits for you and your employer to be successful as a remote engineer and enjoy those benefits you will need to:

Write clearly. You'll write email, bug reports, chat room messages, wiki pages, and a dozen other forms of text all day every day. You need to write clearly and concisely so you can get your point across and avoid misunderstandings. You need to organize your thoughts in a way that other people can make use of them.

Take criticism well. You have to be secure enough in yourself to not assume the worst. Imagine another developer on your project drops you an email that starts "Question about your commit last night." Karl Fogel, the author of Producing Open Source Software: How to Run a Successful Free Software Project, asks, "What is your immediate emotional reaction? If you are worried and frightened, this is probably not going to work out well. If you are curious to know what's in that mail then you're pretty safe."

Learn conventions quickly. Every team follows a set of unspoken rules. These are the small processes that keep the team running smoothly; code style, check in comment formatting, how bugs are referenced, and many others.

> Fogel says, “It’s all on you to find [these conventions] because the effort of saying it to you is so much higher when somebody actually has to type an email and express this thing they’ve never expressed in words before... It is so much work for the other people that they just won’t bother.”

Be self-sufficient. Being in a distributed team means you need to send an email or pick up the phone to get a question answered. This makes communication more difficult, so you need to communicate wisely Put more time into finding the answer yourself. Save the team communication for the times that it helps the team and not just you.

Stay motivated. Working from home means less distraction and more flexibility, but it also means nobody watching over your shoulder making sure you get your work done. You have to stay motivated and get your work done.

Show your work. Much like high school math class: you can’t just have the right answer, you need to show how you got there. Your work can’t speak for itself until it is done. If you wait that long you will have already alienated your team. You have to communicate with your team about your process so they can know what you’re doing before you’re done.

Communicate your status. Managing your status is a way of managing expectations. In the office you are there or you aren’t. It is really easy for your coworkers to see if they can ask you a question. Out of the office you need to let them know when you are there and when you aren’t. You need to make it as simple as possible for someone to see if you are available by using IM, IRC, or another technology.

Be consistent. People are naturally suspicious of what they can’t see and being remote means they can’t see you. Help your team by being a steady presence. Keep consistent hours and let them know you are working hard.

This all might seem like a lot to know. And it is. And it isn’t even the full list. Being a good engineer out of the office is more difficult than being a good engineer in the office. Being remote has huge benefits, and these skills are the cost. The good news is that these skills will help you be a better engineer out of the office or in it.

Every skill that makes you a better remote engineer makes you a better engineer. Every change a team makes to work with distributed engineers will help the team. A distributed environment is like a lens which will focus all of your attention on how the team functions as a team. You can clearly see issues which would be difficult to see in the office. This is just one of the many reasons why working remotely is not a perk.

The Plan of The Book

This book is organized into four sections: getting a remote job, team organization, communication, and balance.

Landing a remote job shows you how to transition your current job into a remote one with convincing arguments for your boss and transition plans you can use with your team and your family. If you are looking for a new job this section tells you how to market yourself, interview well, and land the remote job you want.

Team organization shows you a new way to think about team communication. It gives you an understanding of how to stay close with your team and the tools you need to manage up and improve the way your team works. It finishes with a real world example of a successful distributed team.

Communication is what working remotely is all about. This section discusses the specifics of email etiquette, Internet chatting for business, and the dozens of other technologies you use to communicate with your team. It shows you how to improve your remote presentations and software demonstrations and concludes with an analysis of the common remote problems and how you can fix them.

Balance is the way you stay happy and productive when you live in the office. It contains detailed information for establishing your home office, working well from home, and making sure being on a distributed team works well for your family.

You want to work remotely. And you want to be a better engineer. Being a better engineer means improving your technical skills and becoming a better programmer, but it also means becoming a better team member. The tips, principles, and examples in this book will make you smarter about how teams communicate and that will make you a much better engineer. Understanding how to work remotely will make you a better engineer even if you never leave the office.

SECTION 1

LANDING A REMOTE JOB

Market Yourself Into A Remote Job

There are thousands of brands in every super market. Each box is trying to leap off the shelf and get your attention. If you have ever sat and stared stupefied at the hundreds of options for chips or soda you have a small idea of the difficulty hiring managers face.

This is the problem Michael faces when hiring a new remote engineer. Michael[1] is a development manager for Sun Microsystems working in a European city. He has three remote engineers on his team and they are all great engineers who handle working remotely without a problem. But Michael doesn't want to hire new remote engineers.

Hiring remote engineers is a big risk. Michael wouldn't know what they were doing or if they knew how to work remotely. How would you convince Michael to hire you as a remote engineer? You could tell him you're a great engineer, but every other engineer says that. You could show a strong resume, but many other engineer have strong resumes. If you want to stand out you need to brand yourself.

In the super market you buy Kellog's Cornflakes because you know them. You know what you are getting. Kashi Crunch might sound good, but Cornflakes are dependable because you know them. Market yourself to Michael, and other hiring managers, by becoming a known quantity. The focus of this chapter is making potential employers stop thinking of an engineer in general terms and start thinking specifically about you. The first part of the process is making someone know enough about you to want to learn more. Offering you a job comes later.

Before someone can want to talk to you they need to classify you. You are already helping them do that with the word, *engineer*. With that simple word potential employers know you aren't looking for a job in marketing, sales, or accounting, but it isn't specific enough. There are a lot of engineers.

The first three chapters of this book are all about finding your next remote job. It starts with marketing yourself and making it clear to potential employees who you are and why they should hire you. After you have created a brand for yourself you can find your next remote job. The last chapter in this section shows you how to transition an in the office job to a remote one. It is all about giving you the opportunity to work remotely, and that all starts with getting to know you.

Getting To Know You

A manager is much more likely to take the risk of hiring you work in the office. Watching you work and seeing how well you interact with the rest of the team is much easier when you are all working in the same location. Your potential employer doesn't know if you are a good communicator or a black box. This section will look at different ways of reassuring potential managers by helping them get to know you.

Build your personal network

A personal recommendation trumps everything else. Having someone your employer knows say you are the right person for the job counts for more than

1 Michael isn't his real name

your resume or past job performance. When I interview for a job most people start by asking me, “who do I know who knows you?” The set of people who I could give as answers to this question make up my personal network.

By the time you are looking for a new job it is unlikely a new contact will help you. You need to cultivate relationships throughout your career. Every person you meet could be the one who helps you get your next job. The more people you know the better the odds are that one of them is your common link with your next employer.

Work with more people

This part is just a numbers game. Work with more people and those people can recommend you. Within your company work with other people in your group and other groups. Don’t just stick with your friends.

Work with people in other departments. Those guys in marketing are actually good at marketing. They will also expand your network faster than most engineers.

Work with other companies. A chance to talk to customers is a bonus. Take every opportunity to find out how they are using your products. Take a special interest in them and they will remember you.

Work with other projects. Open source can be a big help here and we’ll talk more about that in building your personal brand.

Cultivate relationships

You aren’t going to meet everyone you need to in your office or online. Go more places where engineers congregate. Travel. Go to conferences and training programs and get to know people while you are there. Talk about yourself and learn more about them.

Every single person you meet could help you later. Take a little time to get to know them. Show an interest in who they are and what they do and they will likely do the same for you. Go out of your way to impress everyone.

Well… not everyone. You only have a limited amount of time and some people really are more important than others. If the CEO of an exciting startup wants to have lunch you can prioritize them higher than a junior engineer. Just remember that everything changes and the junior engineer you know today might be the senior architect you know in five years. Prioritize people when you have to, but don’t dismiss anyone.

Keep your relationships going

When you meet someone and get them interested in you they become part of your personal network. If you go long enough without contacting them they will leave your network. Facebook and LinkedIn are dedicated to helping you solve this problem. An email every now and then goes a long way too.

When you do contact someone make it personal. Show them you were thinking about them. Sending all of your friends a bulk email will make them more likely to lose interest in you.

Creating your network is not something to do overnight. It takes years to meet people and build relationships. Plan ahead and create options for yourself before you need them. Get out and meet new people in the industry. Whenever you meet someone new take a moment to get to know them. Once you have met someone maintain the relationship. The software industry is a small world and you never know who you will run into in the future.

However, you can't always have a personal recommendation. There are a few other models you can follow to help companies find out about you and gain enough trust to offer you a job.

Many companies fear remote workers will just disappear.

The Big Brother Model

Many companies fear that letting you work remotely means you will just disappear. They won't know what you are doing and won't be able to contact you. Fundamentally, they worry that you won't do any work.

oDesk runs a community connecting qualified programmers and web designers with companies looking to hire them as contract employees. They have found a novel solution to this problem. They make a remote contractor a known quantity by thoroughly monitoring the contractor's work habits. Everyone who works for them has special software that monitors all of their keystrokes and a web cam that takes pictures of them at six random times every hour. The amount a worker gets paid is based on this information.

The oDesk method shows a fear of bad communication taken to an extreme level. Instead of risking bad communication from a human (the employee), oDesk ensures a decent level of communication from the employee's computer. This type of communication will never be as useful as good interpersonal communication, but it is much more useful than bad or absent personal communication.

You can apply a milder version of this "no risk" method to your work. If an employer is reluctant to hire you as a remote employee, offer to perform a short project for them as a contract employee. During that project show them that you are an excellent communicator by giving them frequent and useful updates on your status.

Hiring you to work remotely is a scary prospect because employers don't know you. They don't know how hard you work or how good you are at your job. You can tell them, but it works much better to show them. Differentiate yourself by:

Specialization

The best way to stand out from the crowd is to have specialized skills. When you are interviewing remotely you are probably competing against a group of people who will work in the office. Being demonstrably better than those people is difficult. Take advantage of a little marketing wisdom: different is better than better.

Your resume says "Java"? So do many others. Java is too big and too vague to help you differentiate yourself. Be specific. Give an employer some reason to look at your resume instead of the others. In Understanding Successful Blogs we'll look at some examples of personal branding that show you how to differentiate yourself from everyone else.

- Building your personal network.
- Cultivating specialized skills.
- Letting companies try before they buy.

Personal recommendations and the try before you buy model help you get known on the individual level. They are all about making direct connections between you and potential employers. The other ways of helping people know who you are work on a macro level, starting with building a personal brand.

Building A Brand

You're being marketed, whether you realize it or not.

You are being marketed, whether you realize it or not. You are referenced on websites. Other people talk about you. Search for your name and you will find a lot of people talking about you. Every time someone writes about you or talks about you to a friend they are contributing to the perception other people will have of you. The sum of all this information is your brand. If you don't say anything about yourself then other people will have total control over your brand. Letting other people take control of your brand is a big risk.

Being your own marketing department sounds like hype, but you are probably doing it already. When you talk about a project you are working on, describe a tough problem that you solved, or help someone else solve their problems you are building your personal brand. You are informing one more person about you and what you do. Here is a simple example:

Alex Was Hired for Warping Bill Gates

> In 1997, Alex Rosen was applying for a job at a small startup company. The company was looking for developers to work in a brand new language called Java. He had a strong programming background and a good job history, but he wanted to show companies a little bit more.
>
> Before he interviewed with the startup, Alex created a simple Java applet for doing drag-and-drop image manipulation called AlexWarp. He demonstrated the tool by letting users warp a picture of Bill Gates. This may seem like a simple example, but it showed a deep understanding of the Java 2D libraries at a time when Java was still a brand new language. This simple example made Alex's skills tangible.
>
> Alex created this application without the specific purpose of looking for a job and he hadn't even heard of the startup when he did it. The company found Alex because of this applet. They were searching for "Java applet" and found it on their own. At that time writing Java applets was enough specialization to make Alex stand out. The company first heard Alex's name because of the application.
>
> They didn't hire a "senior software engineer," they hired Alex Rosen, the guy who made that Bill Gates thing. It branded him and made him easy to talk about. And they hired him because of it.

Always be thinking of new ways to let people know about you. Build your brand by becoming involved in more projects. Open source is a good place to start.

Working with open source projects

Mark Horstman is a management consultant, not a programmer. He knows recruiting, not code. His advice to engineers is "find an open source project and work on it!" Open source has become a real business force and you can leverage it to help your career.

Open source companies hire a much larger percentage of remote engineers. When you ask those engineers how they got the job, many of them tell the same story:

From Volunteer To Paid Engineer At Mozilla

By Benjamin Smedberg an engineer at Mozilla

I started submitting patches to Mozilla in late 2002 or early 2003. I was just looking for something fun to do, so I was patching. I wrote a couple of patches in the low level [systems], in the Mozilla XPCom object model, C++ templates, hashtable classes, wrappers, and then started getting more involved from there.
At that point Mozilla was just starting out as its own independent entity with a little bit of money —actually hiring people. They offered me a job pretty quickly.

This story fits a pattern I hear over and over again from many open source engineers:

1. I found an open source project that interested me.
2. I made patches and become part of the community.
3. They offered me a job because they knew me and my work.

This model allows you to become a known quantity before the company invests in you. When Mozilla hired Ben they already knew from experience that he was a good engineer and a good fit for the team. Building a history was especially important for Ben because he had no other history in the software industry. You can apply this open source model to corporate jobs too.

Potential employers want to know if you can do the job. The problem with working on proprietary software is that potential employers can't really see what you did. They don't know if your code is well written or barely functional. They can only take your word for it.

The solution to this problem is open source. Contributing to an open source project will help the open source community, but it will also help you. Showing open source examples of your work is much more convincing than any description of proprietary software you could give in an interview.

Choosing an open source project

Choose an open source project that you care about. Don't spend your time working on a project if you don't believe in it. It is more important to work on a project you like than one that is high profile.

You don't have to choose a famous project. You don't need to work on Linux or the Apache HTTP Server. You may have never heard of Mercurial, Pentaho, or Scribus, but they are all popular open source projects with sizable communities. In open source, being a bigger fish in a smaller pond can help you stand out.

Small to medium sized open source projects give you more of a chance to make a difference and show off your skills. Smaller projects also tend to be more focused which makes them easier to talk about. It doesn't matter if a potential employer has never heard of the open source project you are working on. An open source project can be a useful place to show your skills as long as:

1. The project is easily found using Google.
2. You can give a comprehensive description about what the project is, who will use it, why it is important, and what you did for it.

Getting started with open source

There are many open source projects that need help. They are looking for coders, but they are also looking for testers, documentation writers, and translators. Any of these tasks will help you build your brand. Being a good member of the community is more important than being a genius coder.

The best way to start working with an open source project is to become active in its community. Find a project you are interested in and join the users and developers mailing lists. Watch the list discussions while you figure out how to build the project and get familiar with the code. When you are ready, let people know that you want to help. In most projects the right way to do this is the developer's mailing list. Most projects have a list of bug fixes, enhancements, and other tasks they need completed. Pick an item from the list that suits you and give it a try. Repeat the process a few times. If that works out well and you are feeling ambitious you can start your own open source project.

SourceForge.net

SourceForge.net is a free hosting site for open source projects. They provide version control servers, a release mechanism, and a web site for your project. SourceForge gives you an open source project with far less management overhead.

Starting your own open source project

It isn't difficult to start an open source project. Your project doesn't need to be another Linux; find a tool that you wish existed and make it. Create a focused and stable open source project you can talk about well and you will impress your next interviewer.

Recruiters love engineers who have started their own open source projects. An developer who starts a project from scratch shows that they are motivate and organized.

Working on an open source project with other people gives you an opportunity to show you can be a productive member of a community. It will also make you a better developer. Putting your code out for other people to see will make you more careful with what you write and having peer reviews will help you increase your skills. And it builds your brand.

Becoming Publicly Visible

Being a brand and working on open source projects are all part of making yourself publicly visible. It isn't the only way. Sometimes a job offer will come from the places you least expect.

Bad Code For A Good Job

By Jason Orendorff
an engineer at Mozilla

It's kind of a funny story how [Mozilla] found me. I didn't know this until I was already hired, but apparently at the end of a long day, someone was complaining to Bret [Reckard] about some code that was so bad, it was like someone intentionally made it impossible to read. Afterwards Bret randomly Googled for "code that's impossible to read," or something like that, and found the International Obfuscated C Coding Contest (ioccc.org). I was one of the winners back in 2001, and my email address was on the site. That was my in.

...Yeah, they do things a little differently at Mozilla. For what it's worth, that's not the worst reason I've heard for offering someone a job.

Programming contest are lots of fun and helpful. You can also answer programming questions on sites like http://www.stackoverflow.com. However, the place where you have the most control over your visibility is a blog.

I know you've heard it all before. Open source isn't new. Contributing to free software may be good karma, but its certainly good business. Open source helps you:

1. Build your personal network.
2. Build specialized skills.
3. Learn patterns you can use in corporate software.

Open source projects will help you build your brand, but they aren't the only way to get your code or your ideas noticed.

Showcasing Your Ability With A Blog

Being a superstar developer is much more than being a strong coder. A blog gives you a forum to show your other skills. It also lets you tap into the enormous network of tools and search capabilities available in the blogosphere.

Programming isn't just finding solutions; it is defining problems and communicating about them. Good software engineers distill clear and simple problems from complex sets of requirements and user feedback. However, just saying you have this ability is meaningless. Show potential employers examples of how you can define clear simple problems and solve them. Your blog is the place to do that. Let's look at some different types of articles you can write.

Types of blog articles

There are as many types of blog articles as there are types of writing. They range in length from one sentence long updates to novellas. When you are getting started it is worth sticking to a preset list of article types. As you gain more experience with the medium you can expand into other areas.

Back up the buzzwords in your resume with an *I really know what I'm talking about* article. Don't say you know Java, write an article about Java threads. Don't say you know Ruby, write an article discussing the merits of your favorite Ruby design patterns. Your blog is not the place for you to write code, but it is a good place to write about the problems you can solve.

Use a *real world problem I solved article* to show examples of something you did well. Show a little code and talk about why it works the way it does. Make it clear that you know how to ask questions, define a problem, and then solve that problem. Let potential employers see how well you can talk about your work.

How Often Should I Update My Blog?

Your professional blog does not need frequent updates. It isn't like a news blog which needs updates every day. You aren't trying to build a long list of followers, just make some more examples of your work available. Writing a new article as little as once every six months can be enough.

The *here is something I can teach you* post gives you a chance to show off your communication and teaching skills. Choose something simple and explain it. The goal is to communicate information, not impress people with your coding abilities. It is also an opportunity to show an employer that you will be a benefit to a team beyond the code you write.

The last type of article to consider is the *quirky and a little funny article*. This article has a big risk, but it also has a big payoff. If you do it well you can show off your writing skills. It also has the possibility of bringing a lot more traffic to your blog. If you miss the mark you will have something unprofessional and detrimental.

So how do you stay out of trouble with a quirky article? Stay away from the following:

Controversial topics. This includes any topic you wouldn't want to talk about with a room of your coworkers. At the very least this includes sex, politics, and religion.

Personal details. Sharing a few small details is OK, but getting too detailed is unprofessional. Don't say anything you wouldn't be comfortable telling your coworkers in an office, and probably not even that much.

Children and pets. Your family and friends may be interested in that cute trick your new puppy learned or how many diapers the baby goes through in a day, but employers are not.

And your articles have to be good. How do you make your articles good? Start with quality over quantity.

Quality over quantity

A common piece of blogging advice is just write more. This advice does not apply to your professional programming blog. One really good article is worth 20 mediocre ones.

Put your focus into a few good ideas and write them well. The purpose of your professional blog is to build a small group of strong professional contacts, not a large group of readers. Well-written articles are the start of building a small group of interested readers.

Getting readers

All your good blog writing doesn't mean anything if nobody reads it. The first purpose of your blog is to supplement your resume. Give someone who already knows about you a place to find more information. Achieving that purpose is simple: put your blog URL on your resume.

Letting other people find you through your blog is a much larger issue. Remember, you aren't trying to make money with your blog nor are you trying to become a pundit or a public personality. You just want to increase the chances someone will stumble on you. There are three websites to look at for advice about popularizing your blog:

ChrisBrogan.com has specific advice about making your blog better and more useful. He organizes it all into a section called My best advice about blogging. This will show you how to organize your blog and make it understandable to readers.

Copyblogger.com is focused on helping you improve what you write in your blog. How you say something is almost as important as what you say.

Problogger.net has general advice for improving the marketability of your blog.

Understanding successful blogs

The best way to learn about blogging is to study other people who have done it well. Let's look at two examples that will show you how to build your own brand with a blog. These two blogs are very different, but they both make the brand clear and give a lot of useful information to make the brand substantial.

Specificity

John Resig is a JavaScript evangelist for the Mozilla Corporation, the creator of the popular JQuery JavaScript framework, and the author of Pro JavaScript Techniques.

John blogs at http://ejohn.org. He says, "I use my blog as a way to build my reputation." His blog is a good mix of personal branding and useful information. It also makes it crystal clear who John is: a JavaScript guy. JavaScript by itself isn't specific enough, so John further qualifies himself as a JQuery guy. In John's case he is the JQuery guy.

John often posts about small open source projects he is working on. He recently posted "JavaScript Pretty Date" and "Flexible Javascript Events." Each one took John only a few days to complete. These types of real and easy to understand projects are just what employers are looking for.

ejohn.org is successful. It has a large following because John provides useful and focused information. It also creates John Resig as a brand. Type "John Resig" into Google and John's blog comes up first. His blog gives you a good understanding of who he is.

Any potential employer would see John's site and immediately start to feel like they know him. There is enough personal information to be meaningful and enough professional information to make you trust John as an engineer. His site is so effective at showcasing his talents that he had to add a note letting people know, "I'm not currently looking for work."

Personal Branding

Garr Reynolds is a master of personal branding. His message of "presentation zen," including a website and book of the same name, have made his name synonymous with good presentations.

Garr gave a presentation at Google in March of 2008 which you can find linked from his website http://www.presentationzen.com. It starts with a personal introduction.

Garr has built his personal brand well enough that it is easy to communicate and compelling. After he talks for five minutes you know who Garr is, what his background is, and that he is *the* presentation guy. His ability to seamlessly blend his own story with the content of his presentation is an excellent example of personal branding.

Garr isn't an engineer, but his message is important for engineers. You will have many opportunities to talk about yourself. Decide how you want to describe yourself and make sure you can give your own personal introduction in five minutes. The ability to talk about your brand well is an important part of finding a job.

Hiring managers look out upon a sea of engineers with identical skills and need to pick the superstars from the duds. Make their job easier. Stand out and:

1. Take control of your own message.
2. Show off your skills.
3. Make it easy for people to find you.

Your blog represents you. It is your chance to show what you can do on your own terms and it is an important piece of how an employer will find you.

Your career will grow over many years and a number of different jobs. Build your reputation and be a public presence from early on. It will make a big difference when it comes time to get that next job interview.

Land The Remote Job You Love

Building your personal brand, expanding your network, and making yourself publicly visible will start resulting in job interviews. If you've already landed an interview, skip straight to Interviewing. If you're still looking for that interview it's time to make sure you present yourself the right way for that remote job.

The first step to finding a remote job is your resume. Most of the time someone reading your resume already knows your name and just wants to know more about you.

Looking For Your Next Job

The previous chapter showed you how to increase the chances that companies come to you. Now we'll look at ways to approach companies. It all starts with your resume.

Building your resume

We've already talked about the need to be better than a candidate who will work in the office. Your resume needs to be better too. It needs to look more professional and be more impressive than the next guy. Listen to the Manager Tools podcast Your Resume Stinks! from Mark Hostman. Those resources have good advice for a general resume, but they aren't enough. There are specific tips you need when looking for a remote job.

You're an employee first and a teleworker second

When you are working on your resume you may be tempted to include the word *remote*. Don't. Don't list yourself as a remote engineer. Don't say you have skills working remotely. Don't mention that you are looking to work remotely. Just don't do it.

Potential employers read your resume to find out about you and how well you will work in their company. It is all about seeing if you meet their needs. The company does not need you to work remotely. Focus on how you can help them and leave *remote* out of it until later.

Mark Horstman has worked with corporate recruiters for many years. He shares this cautionary tale about mentioning *remote* too soon.

> Mark tells this story.
> Suppose you are in Raleigh, North Carolina in Research Triangle Park. You want to work for IBM [which is headquartered in New York], You send a resume and you say you really want to work remotely. Unbeknownst to you, your resume gets forwarded to the chief of services at IBM in Research Triangle Park. He gets your resume and says, "It's too bad he wants to work remotely. How cool would it be to have his experiences in our office, 10 minutes from his house, and I don't want to hire a guy who works remotely."

This doesn't mean there is no place on your resume to mention working remotely, but don't make it the first or second thing. If you have done jobs well and you did them remotely then you should mention this on your resume. Mark suggests a line in your resume like, "Provided all key deliverables for project X on time and under budget, while working remotely."

Listing jobs that you did well, remote or otherwise, is good. Letting a potential employer know you did the job remotely is good. Saying that you require a remote working environment in your cover letter or your resume is not a good idea.

Specializing to Find your next job

So your resume is solid and it's time to find the right job to apply for. This process starts with specialization. We already looked at some ways to focus your skills in the last chapter, now let's apply that focus to your job search.

Specialization will help you narrow your job search. Take some time and figure out what specific skills you have. Web designer isn't enough, but user experience expert is getting there. Find the skills you have that are different from other people and develop them more.

Once you have a list of specialized skills find companies that do what you are good at. Don't worry if they have job openings right now, just focus on learning the area. Who are those companies? Who are their competitors? What do engineers with those specialized skills call themselves?

Use this information to refine your resume. Tailor it for each company you are applying for. Look for other engineers with those specialized skills, find their resumes online, and make your look like theirs.

Take your newly refined resume and apply to the companies who need people with your specialized skills. And don't just email your resume and wait.

Working with recruiters

Julia Cort specializes in placing remote engineers, and that makes her unusual. Most recruiters focus on knowing the companies in a specific geographic area. In Boston, where I live, many recruiters base their business on knowing the landscape of tech companies in the area. Silicon Valley is the same way. Steve Kasmouski is a recruiter in the Boston area and he almost never deals with remote engineers. Both types of recruiters can help you.

Julia is a big fan of the specialization message. If your resume has skills that others don't it makes you much easier to sell. She will work with you and try to find skills that you can specialize in. She also has a good feeling for what skills companies are looking for at any given time. Recruiters who can focus on finding you a remote job are exactly what you are looking for. The only problem is there aren't many of them.

Recruiters like Steve are much easier to find. Steve works for a Boston area firm, The Winter, Wyman Companies. He has been a recruiter for over 15 years and works with remote engineers less than one percent of the time. He can help you find a job too, but it will take a little work and a little selling.

You can find recruiters like Steve in any major city in the United States. Just Google search for *tech recruiter* and the name of the city you are interested in. Contact the best looking ones and sell yourself. Show them your strong resume, your passion for software development, and your specialized skills. Impress them. Just remember, a recruiter doesn't really care how good an engineer you are. A recruiter cares if they can place you at a job. Make it clear that you are professional, capable, and hireable. Then mention working remotely.

Sell yourself

As soon as you mention working remotely the relationship will sour a little. The recruiter will wonder if it wouldn't be easier to place someone else willing to go into the office. Make it clear to the recruiter that remote is just your preference. Tell them you understand the difficulties in placing a remote engineer and are confident you can overcome them.

After that you just need to sell yourself. Go back to the previous chapter and look at all the ways you defined your brand. Tell the recruiter about them and convince them to show you to a couple of companies. With some hard work and a little luck they will connect you with specific companies and you can start the interview process.

You're an engineer first and a remote engineer second. Lead with your skills, show people why they should hire you, and then talk about working remotely. Find your next job by:

1. Make your resume better than other candidates who will work in the office.
2. Look like the people the company has already hired.
3. Let recruiters help you find a job.

Interviewing

Many books, have specific advice about the subtle art of interviewing. Read one of those other sources, and then apply this specific advice to interviewing for a remote job.

The interview is all about the company's needs. Don't talk about working remotely the minute you get in the door. Save that for the next part of the process: accepting the job.

Remote interviewing has two parts: the phone screen and the in-person interview. Most companies want to talk to you on the phone before asking you to travel for an in-person interview.

The phone screen

The phone screen is not just a hurdle to the real interview. It might feel more relaxed, but it is a big part of the real interview. Do everything you would in a normal interview. That includes physical cues like smiling and making eye contact. When you smile more it makes you more enthusiastic and changes the way your voice sounds. When you look someone in the eye it makes you more interested and that comes through in your voice. It also raises your chin which will open your chest and help you ennunciate more clearly.

So... how can you look someone in the eye when you are talking on the phone? You fake it. Find a good quality picture of a face close up. Something from a magazine can do well. Choose one that is about the same size as a real face. Our brains are wired to process faces of a certain size. Put it on the wall in front of you and look the picture in the eye. It sounds goofy, but it works.

Eye contact is also good because it helps you stay focused. Phone screens cause most people less anxiety than face-to-face interviews. You don't have to worry if you have food in your teeth or sweaty palms when you shake hands. That can be nice, but it is also problematic.

A little anxiety helps you focus which makes it clear that you are interested in what the other person has to say. If you are less anxious, don't let that make you less focused. Don't walk around or do anything else to make noise and give the impression that you are multi-tasking. If the interviewer hears you typing in the background you're in trouble.

You can also help your focus by having your key documents ready. Print out your cover letter and resume. Put your big accomplishments on a separate page with easy to read type. If there are specific ideas you want to get across give yourself a big bold note to make sure you don't forget or having to go looking at crucial moments in the interview.

You are trying to convince the company that you are an attractive candidate and worthy of flying in for an interview. Your ultimate goal is to convince the company that they should give you an offer and the phone screen is a big part of that.

When to talk about working remotely

Talk about working remotely before you get on a plane

Talk about working remotely before you get on a plane for an interview. Working remotely really is a showstopper for some companies. Letting them pay to fly you in and then telling them you must work remotely is rude and will give you a bad reputation. It could even cause a company that would have let you work remotely decide not to hire you because you were deceptive.

This is a bit complex. If you mention working remotely too early you can miss out on your chance to convince them it is a good idea. If you wait too long a potential employer may feel like you were holding something back. The first time you mention working remotely, just try to feel them out. See if it is a showstopper or something they are willing to talk about. Once you know that working remotely is a possibility you can negotiate the details as part of accepting the job.

The in-person interview

Even if you will be working remotely, you almost always need to meet the company face-to-face before they will hire you. This means taking a trip to a different city, talking with people you have never met, and asking them to judge your professional worth.

Be prepared. Mark Horstman says, "Your job is to know your background well enough---and to be so skilled at delivering it through a series of preparatory processes of practicing the big questions you are going to get in an interview — that when you get interviewed you can answer any question that someone throws at you, whether they are HR, IT, or a software engineer."

Be excited. "The best way to convince an HR representative that you can work well remotely is to have the right background results and a very persuasive, engaging, friendly, warm, smiling (even on the phone), interview."

Dress nicely. Many offices are casual, but it is still a good idea to dress well. The interviewee should be better dressed than the interviewer.

Ask questions. They are interviewing you, but you are also interviewing them. Asking questions will also give you a little more control over the interview. A question is a gentle way to guide the interview in the direction you want.

Answer questions. The interviewer will ask you questions. Answer the question you are asked, not the question you wish you were asked.

Listen. Talking is important, but so is listening. Give people the chance to talk about themselves and the company.

Take notes. Bring a paper notebook rather than a laptop. A paper notebook lies flat on the table and doesn't interfere with your ability to interact with the interviewer. A laptop sits in front of you and gets in the way of interaction.

Ask for names and titles. Write down the name and title of every person you talk to. Ask where that person fits into the company, and take their business card.

Be confident. Or learn to fake it. Interviews are stressful, but you must do your best to appear confident, calm, and collected. Arrive early so you aren't rushing, get plenty of sleep the night before, and take a moment to breathe.

Talk about yourself. The purpose of the interview is to let potential employers learn more about you. Help them out by talking about yourself. Tell stories from your life (make sure they are safe for the workplace). Taking the time to talk a little more will help the other person get to know you.

Never decide you don't want the job during the interview. You can't hide your discontent from the interviewer if you make up your mind that you don't like the company during the interview. Delay any decision until after the interview is over and leave on the best possible terms. Remember, the person working at the company you don't like may be hiring for your dream job later.

Be polite. Make sure to look people in the eye, give them a good handshake, and say please and thank you.

All of these steps are about one thing: getting a job offer. Once you have the job offer you need to accept it and that is a more complicated than just saying *yes*.

Accepting The Job

After the company makes you an offer, the focus shifts. You are done convincing them to make you the offer and you can begin the process of accepting the offer. This is the part of the interview process where you can negotiate salary and other benefits. It is also the place to talk about the details of working remotely.

When you mention your desire to work remotely you need to spin it positively. Never lie, but focus on how it will work well for the company. Don't present it as a detriment. Talk about the opportunity the company has to work with you remotely.

Show them that you have thought through all the issues of working remotely. If you have done it before talk about what made it work. Share the negatives along with the positives. It makes you sound more informed and therefore more convincing. Set their expectations and negotiate the specifics of how you will work remotely. Make sure to answer the following questions.

Plan the first two months. Will you work in the office for a while or be remote from day one? Are there training programs you need to attend?

Work out your travel schedule. How often will you come back to the office? Is there a fixed schedule or will you travel based on specific project needs?

Examine any changes the team needs to make. Do they already use some form or real-time chat technology like IRC or IM? Will they have to buy special equipment like conference room phones?

Discuss your expected working hours. Does the company have flexible hours? Are there specific times of the day you need to be available?

Ask about performance metrics. How often will you have performance reviews? How will they judge your work? Most managers don't like to admit it, but the number of hours you spend in the office counts for your performance appraisal. How will you replace that?

Come to an agreement about home office expenses. Will the company pay for your Internet connect, telephone, or office equipment?

During this entire process stress that remote does not mean invisible. You want to meet the team face-to-face, you are going to travel to the office, and you will be available throughout every day.

Getting a new job always takes work and causes stress. Take control over the process by being proactive. Don't wait for companies to find you. Go out and find them.

Don't ever make "remote" the first thing you say or the last thing someone remembers about you. The fact that you want to work remotely is just a detail. The company will decide about hiring you based on how well you present yourself and how well you fit into the company.

Finding a remote job can be difficult, but with a little preparation, persistence, and luck you'll find a new job. Or you can transition your current job into a remote one.

Transitioning To A Remote Job

When Jim Lemke made the case for working remotely to his boss at Redhat it was simple. He was a valued employee and had built up a strong track record with his team and his boss. He moved away from the office and started working remotely half-time. After a while he transitioned to full-time remote.

Jim's story is notable for how unremarkable it is. I hear this story from many people. Compared to finding a new remote job transitioning is easy. You are already a known quantity and in a much better position to make the case for working remotely.

If nobody else in your company, department, or group works remotely you need to answer a larger question first: why should your company let anyone work remotely? You might run up against company policies, the human resources department, or a concern that letting you work remotely will open the floodgates for everyone else.

Working remotely works for your company

After you have laid the groundwork and made it easier to say yes, show your boss that a remote engineer program can be good for your company. Make a simple rational argument that allowing remote engineers is good for your company. The decision may not be up to your boss alone, but they are your first advocate with the higher levels of management in your company.

Commuting 40 minutes each way wastes 8 work weeks a year

Let's start with the bottom line: teleworkers save money. It is all about the bills your company pays to provide you with an office to work in, supply you with a desk and chair, and all the other expenses involved with running a large building. It costs about $10,000 per year just to keep you in a cubicle.

Your company can save a lot of real estate and facilities costs by having more engineers work remotely. IBM, Sun Microsystems, Cisco, and Compaq are all saving tens of millions of dollars every year with remote worker programs. When your boss worries about starting a

remote worker program, remind them of the money they can save. Saving money is good, but making money is even better. Working remotely lets you be more productive. You have more time to work. If you commute just 40 minutes each way every day you are losing eight working weeks a year.

You don't just save time with less driving. Offices are full of people. That lets you have meetings and share ideas, but it also lets you waste time. Getting away from your coworkers gives you more time to work.

Working from home also makes it much easier to work beyond the nine to five. Have dinner with your family and get some work done after the kids go to bed if you want to. You have the added flexibility that lets you to find more time to work.

Supporting remote workers helps your company do well by saving money and having a more productive workforce. It also helps them do good by saving the environment. Many states offer renewable energy credits or other benefits to companies with environmentally friendly policies. A remote worker program can help your company qualify for those benefits by wasting less resources on buildings and substantially cutting down the fuel consumption and greenhouse gas emissions associated with daily commuting.

Once you convince your boss and the rest of your company's management that a remote work program is a good idea in general, you still need to convince them that it is a good idea for you.

Make it easy to say yes

Your manager is the first person you have to convince, and they probably have some reservations about letting you work remotely. Make it easy for them to say yes by addressing some of their basic concerns.

Be trustworthy. The most convincing argument that you will do a good job remotely is to do a good job in the office. Build your manager's trust by performing well. Show them your history of dependable performance and make it clear you will continue that stable performance remotely.

Define performance metrics. Make a plan for assessing how well you are doing once you start working remotely. Ask for extra performance reviews during the first six months. Make it clear that you intend to show tangible results when working from home.

Make a plan for how often you'll visit the office

Make a travel schedule. Have a plan for how often you will return to the office after making the transition. If there are any upcoming events that require you to be there in person, plan for them ahead of time.

Talk about your job, and show your manager how you will do your job remotely. Are there times when you need to physically touch hardware? Do you spend a lot of time in face-to-face meetings? Create a plan for how you will accomplish each part of your current job remotely.

Talk about collaboration. You need to communicate well with your team and your boss needs to understand how you are planning to do that after you leave the office. Detail any new tools you want to use with your team. Will the rest of the team need to do anything differently to collaborate with you remotely? The communication part later in this book discusses the issues of communicating with a distributed team.

Have your coworkers speak on your behalf. They are the ones who will need to collaborate with you and having them believe it will work is a big part of the argument.

Give your boss an escape clause. There are many good reasons to work remotely, but it can be difficult to get final approval. Your boss might worry they can't bring you back once you start. This is especially true if you are moving to a different city. Ease their fears by reviewing what will happen if working remotely doesn't work out, or doing it for a trial period before you finally commit.

Working remotely works for you

Let's assume the advice in the preview sections worked and your company is open to a remote worker program in general terms. The next question your boss will ask is: *Why do you want to work remotely*? You need to have a good answer to this question in addition to the arguments in the previous section. This part isn't about company policy. It is about you. Will working remotely work well for you?

Saving money and extra productivity work better in the discussion of a general program. Try to talk about productivity and you run the risk of your boss wondering why you can't do good work in the office. Focus on the two reasons that are easiest to understand and most difficult to argue against: family and geography. Here are a few family reasons that might help:

- Your family needs to move.
- You want to take care of your small children.
- A family member is ill and needs extra care.

Present solutions, not problems

Everyone understands family, and it sounds a lot better than, "I really want to work in my underwear." Even if your family isn't the primary reason you want to work remotely, it is a good one to start with.

Using your family also makes sure that your request does not appear to be a personal reproach. It sounds more like, "I really want to stay in the office but I can't," and a lot less like "I'm sick of looking at you every day."

If you can't use your family, then use geography. Talk about wanting to move. Make it clear that you are making this transition because it is the right thing for you and it doesn't have anything to do with your boss or your team. Reiterate that you really want to keep doing the work you have been doing and you are looking for a way to make it happen. Present a solution, not a problem.

Once you have shown your boss how working remotely will work for your company and for you, the next step is to make it work for your team. Show your boss you have a plan for making the transition go smoothly with your team. Start by being open and upfront about any changes your team will require to support a remote engineer. They shouldn't have to do much to support you, but there are some required changes.

Most of the changes focus on communication. You need a plan for how you and your team will communicate when you aren't in the office. What technologies will they use? Does your company need to buy a conference phone? Does your team need to start using IM or IRC?

The third part of this book is about the specifics of communication. Take a look at the communication options, figure out which ones are the best for you and your team, and discuss it with your manager.

Working remotely really works and it can really work for your company, you, and your team. Position your reasons as benefits instead of problems. It isn't *we'll have these problems but I guess we can work around them*, it is *here are some solutions to help us all move forward*. Frame it in a positive way and you have a much better chance of convincing your boss and getting to make the transition to remote work. And plan out the following items:

1. Your travel schedule
2. Your job responsibilities
3. How you will handle trouble
4. How working remotely can help your team

Now that you have a plan in place you're ready to make the transition.

Making The Transition

The transition from working in the office to working remotely can be stressful and difficult. Don't expect to find a good balance between work and the rest of your life in your first few weeks away from the team. That comes later in the process, and later in this book. Just focus on the first month.

How you behave in the first few weeks of working remotely sets the tone of your new working relationship with your team. Take the time to explain your transition to them.

Have good reasons for working remotely and share them. Tell your team all the reasons you told your boss. It will help them feel more involved in the process. Your boss gives you permission, but your team will make you successful.

Reassure your team that you aren't leaving them. You aren't working remotely to leave your team; tell them that. Make it as clear as possible that you want to remain an active part of your team.

Make it very easy to contact you. Let your coworkers know you will be only a phone call away and encourage them to call you whenever they need to. Make a point to be even more available during your first few weeks working remotely.

Resolve personal conflicts before leaving the office

Leaving the office is a major transition. Spend some time with the rest of your team before you leave. This may be your last chance to see them face-to-face for a while. Make your working relationships as solid as possible. If you have any outstanding issues with other people try to resolve them before leaving the office.

Hit the ground running. Set up your office, computer, telephone, and Internet connection ahead of time. The balance section at the end of this book has many tips for setting up a home office. Get them sorted out ahead of time so you can start working as soon as your first remote workday starts.

I spent my first remote week glued to my computer, afraid to miss even one instant message. It got better for me and it will for you. Focus on productivity and communication; your first week will be over soon and you can find a routine that works for you in the long term.

During your first week focus on only three things: productivity, availability, and not being too hard on yourself . Create something your first week. It doesn't really matter what it is, but it has to be tangible. Just pick one area and show your team that you are still working hard.

Be available from the first minute of your first day. You will have time to establish boundaries and find a routine later. At the beginning be there for every phone call, IM, and email.

Stay relaxed during your first week. Your team will need a little while to get used to communicating with you in this new way. Don't expect them to get back to you as quickly as they did in the office. They will get better with time.

It takes a few weeks to establish a routine. At the beginning your schedule will be a little erratic. Do what you can to maintain stability, but don't worry about establishing a schedule.

Transitioning to a remote job is a beginning and like any beginning, it sets the tone. You can make your life as a remote engineer a lot easier by planning the transition and executing it well. Focus on the short term at first. There will be time for longer-term planning later. For now focus on:

1. Making your first week go well.
2. Communicating well with your team.
3. Hitting the ground running.

Transitioning from a current in the office job makes it much easier to convince your boss to let your work remotely. Everyone knows who you are and what a good job you do. If you lay the groundwork and plan ahead you'll get permission to work remotely. Then you need to make it work.

SECTION 2

TEAM ORGANIZATION

Team Profile:

CodeSourcery - An Open Team In Action

Open teams are easier to understand in a volunteer project. You need to keep the contributors or they won't volunteer. Do open teams work in a corporate environment? Can they help you manage a distributed team of entirely full-time employees? I spoke to Mark Mitchell and other members of his company Code-Sourcery to find the answer.

Mark Mitchell doesn't like to describe his company as "virtual". It's distributed, but CodeSourcery is very real. If your company produces a new piece of hardware that needs to run the GNU C/C++ compiler, then call CodeSourcery. They solve hard problems, delight customers, and work together without a single office.

CodeSourcery has about 30 engineers located all around the United States, as well as Canada, England, and Russia. Every employee works from home. They develop complex system-level code for new chipsets most engineers have never heard of and create a cohesive team out of a group of people from around the world.

I spoke with Mark Mitchell, the CEO of CodeSourcery, and Nathan Sidwell, the manager of their GNU toolchain development group, as well as team members Ricardo Anguiano, Catherine Moore, and Stephan Seefeld in October of 2008. They discussed what it was like working in an open team and how it feels to work for a distributed company.

Make It Easy To Give You An A

Managing and coordinating 30 engineers in 30 different locations may sound like a nightmare, but Mark makes it work. His management style is based on a simple mantra: make it easy to give you an A.

Mark had a professor at Stanford that started his first session by asking the students what their purpose was in the class. There were many answers: "To succeed." "To learn the material." "To complete the problem sets." After a little while the professor stopped them all and said, "Your purpose in this class is to make it easy for the grader to give you an A."

Making it easy to give you an A means going out of your way to let other people see how good your work is and showing your work along the way. If you deliver a large piece of code that nobody has ever seen before, it is almost impossible to review and accept it. Making it easy to give you an A means showing frequent updates so other people can follow your process. Ricardo explains how this affects his work, "If I don't communicate, if I don't tell people what I have done, I might as well not have done it."

CodeSourcery uses this mantra for internal coordination. They also use it to train their engineers to interact with their customers. Many companies insulate their engineers from their customers, but at CodeSourcery every engineer works directly with clients.

Tracking Productivity

Your manager probably worries about how to track your productivity. They worry you won't work when nobody is watching you. Mark sees these types of productivity concerns as the "I want my people where I can see them" kind of mentality.

This isn't an issue for CodeSourcery. Mark freely admits, "We can't really tell if you are working a 14-hour day, an 18-hour day, or a 6-hour day." Mark cares about the work his team produces, but he is also very careful about letting his team have a good work/life balance. Mark advises his team, "If you are working hard, then say you are working hard." He encourages everyone to openly and honestly communicate their status so everyone can help each other be productive without becoming overloaded.

As managers, Mark and Nathan find that tracking productivity with a distributed team just isn't a big issue. The engineers discuss their status in a weekly meeting, and they talk about their problems throughout the week. Each engineer is also expected to send a weekly status email detailing what they did during the last week and what they intend to do during the next week.

Mark is very clear with his team that he wants to hear, "this part has done well and this part is a little tough..." This involved style of communication makes it easy to assess the contributions of individuals and the progress of projects.

Staying Close Around The World

The CodeSourcery team uses three main tools to stay close with each other: IRC, telephone, and email.

IRC

They depend heavily on IRC. Developers at CodeSourcery will typically remain logged into IRC for the entire workday. Most of the time developers minimize the IRC window and wait for the program to alert them when someone types a special keyword like their name or the name of the module they are working on. They will also check in with what the rest of the team has been saying periodically.

> Catherine Moore:
> "Lots of times I don't pay attention to [IRC] because it can be very distracting. Usually I just join in when I need help or something catches my eye that I could help with."

The CodeSourcery team often communicates sensitive information over IRC. They protect their privacy and the security of their customers by using an internal IRC server.

Telephone

IRC is an important part of the CodeSourcery communication, but it doesn't work well for high-bandwidth communication between two or three people. The team switches from IRC to telephone when they need a more in-depth discussion. Every employee gets easy access to an international conference calling service and picking up the phone is encouraged.

In addition to ad hoc calls the team uses a weekly conference call to share status. This is the chance to hear updates about the overall team and each project. It also gives Mark the chance to make company announcements.

Email

CodeSourcery uses mailing lists to discuss issues, perform code reviews, and share information. They have mailing lists for specific projects as well as mailing lists for the entire company. Email is a part of their communication, but they use it for specific issues and status rather than open-ended conversations.

CodeSourcery also maintains an internal wiki server, but they use email for the bulk of their written communication. Email also provides a good way for a team member to raise issues after hours.

Personal Availability

CodeSourcery is based around a culture of personal availability. Every team member is responsible for being easy to contact. Mark makes it very clear: any person who "went dark" for a long period of time would have their career prospects "severely limited." All employees are strongly encouraged to send emails about their availability and log vacation time in a shared calendar.

Another aspect of personal availability is adjusting working hours to fit into different time zones. Many of the California-based employees work earlier in the day while employees in England work later into the evening. The one Moscow based employee changed his hours to work with the team in England. This work schedule is informal, but the goal is to get a few hours of overlap every day.

Being An Open Team

CodeSourcery builds an open team from the top down. Mark deliberately fosters a supportive environment where team members can share their problems and help each other.

> Mark Mitchell:
> "There's no shame in getting stuck, there's no shame in not knowing, and there's no prize for figuring out something that someone else has already figured out. The point is to work as a team and ask for help and collaborate with each other; to learn from each other and move the overall thing forward."

The team saves a lot of time and effort because of their open culture. They reuse a lot of ideas and build on each other's work. Mark knows that "everyone who works here is smart enough to figure everything out from first principles, but there's no point in doing that."

CodeSourcery realizes that a team is greater than the sum of its parts. When one team member gets better at their job it helps the whole team and when someone has a problem the whole team worries about solving it. From the top down, everyone in the company is inspired to help other people on their team.

> Stephan Seefeld:
> "[Working in a team means] helping others and being able to express yourself if you need help, and apply that help well. ...It is all the more important when you are distributed."

Feeling Like One Organization

CodeSourcery makes every employee feel like a full member of the team. They help each other out and mentor new team members. The team also gathers for an annual retreat every January. This is a time to do intense project planning as well as a time to relax and get to know each other a little better. Nathan says, "To start off the new year with that kind of meetup is a very good start to the year."

During the rest of the year the team is very focused on IRC, email, and telephones. They don't often meet face-to-face and never use video conferencing. Nathan told me a common joke in the company: "We don't know if we've hired a 14-year-old girl who is just pretending [to be an adult] until January." Most CodeSourcery employees are hired without an in-person interview.

The Only Man In The Office

CodeSourcery has no offices, but they do rent out a server room in California. The contents of that server room are the responsibility of Ricardo Anguiano. Ricardo drives the short distance to the server room about once a week, but the entire system is configured to be administered remotely.

Ricardo uses special power outlets which can be turned on and off via telnet. This allows him to shutdown and reboot servers remotely. He also builds custom hardware connections in order to remotely press reset buttons or restart USB devices. Ricardo is the one on-site employee; he keeps everything running smoothly for the rest of the company.

The Benefits

Mark Mitchell is the CEO of a successful small company who eats three meals a day with his wife and young son. Working remotely gives Mark the balance he needs to be a good CEO, husband, and father. Being a distributed team also has many benefits for his company.

The People

Finding someone who can do the type of low level Linux systems programming CodeSourcery requires is very difficult. Being a distributed company allows CodeSourcery to find talented people anywhere in the world. It gives them access to an enormous pool of potential employees. The success of a software company depends on the people they hire. Taking advantage of talented people from all over the world gives CodeSourcery an amazing advantage.

Financial Savings

CodeSourcery spends almost all of their budget on salary. They have very few travel expenses and almost no facilities expenses. They don't pay for office space. It costs an average of $10,000 per year to provide someone a cubicle to work in. With more than 30 people working for them, CodeSourcery is saving at least $300,000 a year. Offices also need receptionists, phone systems, network support, and facilities managers which CodeSourcery doesn't have to pay for.

Home Warriors
The Economist, November 11, 2008

CodeSourcery does supply their engineers with the basics, like phones, printers, and Internet access. But the cost is very small compared to supporting them in an office. They do whatever they can to support their remote employees. Many companies see working remotely as a privilege —CodeSourcery sees it as a competitive advantage.

A Great Job Where You Want To Be

Some people wouldn't leave Manhattan for all the money in the world, while others would pay to stay away from the city. There isn't a single right answer to where you should live. CodeSourcery's distributed team allows their employees to live in a location that works for them.

The flexible hours also make a big difference for the team. Ricardo told me, "I'll go for a run, or I'll go to the gym, or I'll go for a bike ride. It is definitely a perk to be able to do that." CodeSourcery offers their team flexible hours, interesting projects, and the chance to live anywhere in the world. It is a very attractive working environment.

A Fully Distributed Company

CodeSourcery is a distributed company in every sense of the word. They also work as a cohesive team to solve some really tough problems. Being a distributed team is a strong competitive advantage for CodeSourcery in part because it is a big hiring incentive. They find talented people around the world and don't limit themselves by geographic location.

They also cultivate a very open culture. Not just code changes, but almost all company decisions are discussed with the whole team. Engineers are encouraged to help each other and bring up problems as soon as they happen. CodeSourcery proactively creates an environment of sharing and openness.

> Stephan Seefeld:
> "You have to be a good communicator. Being able to communicate is what makes it obvious that you take feedback well. Because you have to show it somewhere. If you just listen but don't speak up and

> don't show that you actually understood what the other [person] was saying then we don't know."

CodeSourcery is an inspirational example of a different way for a company to work. They are also a useful guide when facing your own issues working remotely. Use their open communication as a model when you have communication and coordination problems to solve.

Staying Close When You're Far Away

Being successful as a remote engineer requires being a good team member and being a good team member is about more than doing your job well. It is about creating an emotional bond with your coworkers. You need them to think of you as a human being rather than just a screen name. The emotional bond starts with good communication, but it needs to go beyond that.

The first step to creating this emotional bond is to understand what you are missing by being remote. Technology will connect you with people from around the world, but it doesn't create the same close connection as meeting in person.

You want to get a feel for who your coworkers are as individuals. Get to know something about them. Begin to predict how they will react, what they will like and what they won't. They key is learning enough about someone to know them emotionally. And teaching them enough so they can understand you.

When you meet with someone face-to-face you are exchanging an enormous amount of visual information. Faces tell you a great deal about what someone is feeling and thinking at that moment, as well as a lot of information about that person's background. Most of this information comes on a subconscious level. We are all experts, even if we don't know it.

Being remote means missing out on all of this information. A person can send you an IM about what they are feeling or talk to you on the phone about what they are thinking, but they are only giving you a recreation after the fact and not the whole story.

Psychologist Dr. Paul Ekman states that, "...all facial expressions of emotions are involuntary; they are never voluntary or deliberately made."[1] He also states that, "Even if we asked the person [to explain their emotional state when making the expression,] we would not find out. For she could only tell us what she thinks happened. While that is interesting to know, it is likely to be a retrospective construction, not what actually happened."[2]

Unlike facial expressions, all remote communication is voluntary. People think about what to say (we hope) before they say something over the phone or write an email. The inherent inability of people to recognize their own emotional state makes them incapable of conveying that emotional state to you. It also makes you unable to convey your emotional state to them. This is a serious impediment to human interaction and you will not be successful in a distributed team unless you overcome it.

You can replace a lot of that information through good communication, but you can't replace all of it. Compensate for this loss of information by getting to know the people you work with. Knowing someone well will help you predict behavior and replace some of the information you are missing. You are the remote engineer so you are responsible for going the extra mile to stay close with your team. Staying close with your team will help you get your work done, have your ideas heard, and be more successful.

1 Ekman, Paul *Should We Call it Expression or Communication?* p. 339.
2 Ekman, Paul *Should We Call it Expression or Communication?* p. 335

Staying Close With Your Team

Your relationship with your team —the people you work with every day— is your most important working relationship. These are the people you will spend the most time with and they are the ones who will have the largest emotional investment in you. Your team has enormous influence over your job tasks, your job satisfaction, and whether you have a job at all. Your first responsibility as a remote engineer is to connect with your team.

Start With Face-To-Face

There is no substitute for in-person communication. When you first start working with a new team you should always meet them in person. Many distributed teams have new members work in the office for at least two weeks and often much longer. It is important to start the relationship with as much information sharing as possible. Learn the systems you will be working on. Tell your team what your areas of expertise are. Learn about your coworkers' personal lives and tell them about yours.

During this time you should meet as many members of your team as possible. When you meet someone face-to-face you will create a bond. You can extend that bond to create a more meaningful relationship when you are out of the office. That bond is the beginning of connecting with people on the emotional level required to really be a part of the team.

Your Picture Is Worth More Than 1,000 Words

You must provide a picture of yourself in a public place for your company and your customers. It doesn't need to be a professional photograph. A recent picture with a close view of your face will do the job. For the coworkers you have met, the picture will keep you real to them; for the ones you haven't met, it will help them think of your user name as a real person.

One Laptop Per Child In Nepal

by Bert Freudenberg from Viewpoints Research Institute and One Laptop Per Child

I went to Nepal for 10 days last year because they are using the Squeak programming language for their One Laptop Per Child work. They flew me in to get them jump-started on some of the non-obvious aspects; the stranger parts of Squeak and Etoys. They were so happy that I showed them a few tricks that are not written down in any book. If you are an experienced user of the system [those tricks] make your life so much easier.

If you can watch over someone's shoulder for as little as a week you will pick up so much more than if you're exploring the system by yourself. There are so many little things you can learn by being with someone in the same room.

Face-to-face meetings are important. However, being a remote engineer means most of your time will be spent away from your team. Most distributed teams only meet in person once or twice a year. During the time between in-person meetings you must cultivate personal connections with your team.

Talk About Yourself... A Lot

A short phone conversation and a few emails don't add up to a real person. You need to take the time to let the rest of your team know about you. The big stuff is important --kids, where you live, major hobbies-- but talk about little stuff too. What did you do over the weekend? Have you seen a good movie recently? Did something funny happen to you in the last few days?

Make talking about yourself an ongoing habit. Don't talk about yourself for a hour each week; make it five to ten minutes every day. The early part of meetings is a good time for this. People are normally chatting, just make a point of joining in and tell them something about yourself. Letting the rest of your team know about your life helps to humanize you.

Talk About Other People Too

> "Asking someone a question isn't just a way to get information. It is also a way to build a relationship."
>
> - Atul Varma, Mozilla Labs

Part of asking someone to care about your life taking the time to care about theirs. Ask the other members of your team lots of questions. Don't be pushy, just be interested. Telling you about themselves will help them feel closer and more connected to you. It builds trust. Listening to someone else's stories will also help you get a better idea who they are. It is a lot easier to understand someone's motivations if you know something about them.

Make Time To Waste Time

Many remote engineers don't engage in casual communication. When you want to talk with someone you need to schedule time, plan out an email, or bring up a chat client and type "Hello." This barrier can almost eliminate spontaneous conversations from your workday. You really need to make the time to waste time with your coworkers. A simple way to waste time is bad jokes. Old, overused, and downright corny jokes are a good way to waste time with people you don't really know.

Cultivate Personal Connections

Getting along well with your whole team is a laudable goal, but it is easier to connect with individuals than groups. There will always be a couple of people on the team who you connect with more than everyone else. Find those people and cultivate those personal connections. Having closer relationships with individuals within your team will make you happier with your job and help build trust.

Trust is a tricky thing. Exactly how and when one person begins to trust another is very difficult to define. You need to get your team to trust you, but this a slow and gradual process. Start with just one person. Go the extra mile and give that person a reason to trust you. Then give them a few more. Once that person trusts you they can be your advocate on the team. They will help other people trust you.

Manipulating People Is Good

Trying to gain the trust of one person so they can help you influence the rest of the team may seem manipulative. That's because it is manipulative. You may have been told that manipulating people is a bad thing to do. It isn't. Exploiting people is a good thing, but only if you are open and honest about it.

When you find one person to be your advocate tell them that is what you are doing. Explain that you aren't as close with the rest of the team as you would like to be and you are hoping they can help. Ask them for their advice. Let them know that you hope they will help you improve the level of trust the team has in you.

Manipulating people is seen as bad because it is often nonconsensual. The person being manipulated doesn't even realize it is happening. When you ask someone for their help as your advocate then you are informing them and asking for their consent. Asking people for help is a good thing.

Hallway Conversations

A coworker once said, "I feel like we get more done walking back from the meeting than in the meeting."

The hallway conversation is a very difficult habit for most engineers to break. You may be remote, but you can't ask people in the office not to talk to each other. What you can do is insist that they include you in important decisions. Don't be afraid that you are missing out because of hallway conversations. As long as your team keeps you updated it doesn't matter where they talk.

Cross-Cultural Connections

Being remote means it likely that you will be working with people from different cultures. You may have coworkers from around the world or just another part of town, but they all have disparate backgrounds and cultural expectations. It is always important to be culturally sensitive, but you have to do a little more. Work hard to create personal connections across cultural boundaries.

Getting to know someone's culture helps you start to understand them. Most people are happy to talk about where they are from. Keep your questions very open-ended and let the other person talk for a while. Did you grow up near London? What is it like to live in Provo? Are there many good restaurants in Brisbane? These questions may seem bland, but they are a good safe way to get conversations started.

You should also share something about your culture. Small and noncontroversial details are best. Do you live in the city, suburbs, or country? Do you like to spend time outside, or just play shooting games until your eyeballs feel like they will fall out? Where are you from originally? Have you had other careers in the past?

For example: I live in the city of Cambridge, Massachusetts, but I have always dreamed of living out in the woods. I like container gardening, cooking, and playing Go. Many years ago I spent one doomed night as a French pastry chef.

These small details give you a little piece of the picture of who I am. They also give you places to start conversations. They make it a little easier to talk to me.

Religious and social differences are also an issue. For example, in some areas of the United States it is inappropriate to take the Lord's name in vain and in others a full range of four-letter words are OK in the workplace. You need to be conservative about the jokes you tell or details you give until you understand their background and expectations. Many practices that are acceptable in some cultures are taboo in others.

Along those lines, there are specific controversial topics of conversation you should avoid when talking with your coworkers: religion, politics, intimate relationships, ethnicities, and gender differences.

You don't have to be a boring person. Just be a little careful when you are meeting someone new in a work context. It is better to be a little bland than offend someone.

Don't Just Be Yourself

I'm sure you've been told many times during your life to just be yourself. It is good advice in social situations, but not for the workplace. When you are interacting with your coworkers, especially in the short bursts that being remote requires, just being yourself isn't enough. You should be a kinder, friendlier, more understanding and approachable version of yourself. Take the time to listen a little more than you normally would. Give people easy conversation starters. Remember that the workplace is an artificial representation of real life. You need to be yourself, but a more polished and palatable version of yourself.

Staying Visible

Working remotely makes it difficult to create emotional bonds with your team. It also makes it much harder to stay visible within your company. Doing well within your team is good, but promotions, raises, funding, and layoffs will all be decided by people outside of your team. Being remote means being mostly invisible to the people in your company in different teams.

You have to make sure that other people know who you are and what you do. They will never be as emotionally invested in you as your team is, but should still see you as a valuable member of your company. Being recognized by the executives in your company is important for everyone, and being remote makes it more difficult.

Participate In Presentations

You absolutely must participate in the presentations your team gives. When your team gives a presentation there may be a tendency to give the people in the office a larger role. Giving presentations can be difficult enough without worrying about being remote. You need to speak up and say that you want your fair share of presentation time. The people you present to --other teams, managers, and executives-- will make important decisions in the company. You need to make sure they know who you are and what you do.

Talk Directly With Customers

Most engineers aren't directly involved with sales and marketing. Your opportunities to talk with customers come in the form of product support and feedback. Your company has forums, bug tracking systems, and other places where customers can ask question. You should be active on those forums so you can help your customers, increase your knowledge of how your product is used, and stay visible to other people in your company.

When the opportunity to support customers arises, don't pass the buck. Helping a key customer out of a difficult situation is a great way to earn recognition with the customer and with your company. Helping customers takes time away from being a productive coder, but getting an important customer out of a jam is almost always worth it.

Take every chance you can to hear what your customers think of your products and what they want to see from them in the future. Engaging your customers in a conversation about your products will improve your product offerings, it also creates relationships with your customers. You may be in a different location, but you are part of the company and you want to make your customers know it.

The goal of interacting directly with customers is to create working relationships that make the customers happy. If a customer mentions you by name as a reason they like to do business with your company it will be a big help to your career.

Contribute Outside of Your Team

Many companies have optional groups or cross-functional teams. They decide new coding standards, investigate new project planning methods, and make decisions that will affect teams across the company. Get involved. It helps you meet more people within the company and stay more visible outside of your team. People won't see you in the halls or around the lunch table. These cross-functional teams may be their only chance to get to know you.

Help People From Other Teams

Peer review and code review inside your team is important, but you should look for opportunities outside of your team as well. Other teams may lack your particular expertise or just need another set of hands. You don't need to be doing this constantly; helping other teams a few times a year will help you in the long run.

Attend Corporate Events

Attend all of the corporate events you can. In addition to seeing your coworkers face-to-face at casual events, go to user conferences or important customer meetings. Make sure that you are a part of the big events in your company.

Attend Industry Events

You should also attend industry events outside of your company. Trade shows and user conferences can provide a new place for members of your company to congregate. They are also a good place to build your social network.

Emotional Connections for Professional Success

Being remote is a barrier to the distribution of emotional information. That means you need to go above and beyond to stay close with your team. You can start with an in-person meeting and build on that meeting later with email or your telephone conversations. You must cultivate other ways of sharing emotional information because you won't have as many opportunities for meeting face-to-face.

Sharing emotional information is all about details. Let your coworkers know details about you and learn details about them. Details, important or otherwise, will make a screen name feel like a real person. Make the time to waste time with your team.

It is important to connect with your team. It is also important to connect with the rest of your company. Staying visible to the executives in the company as well as other teams will help you get raises, promotions, and funding. Take the time to participate in presentations, communicate with customers, and help other teams. This extra effort will be rewarded by increased visibility in the company and more support from upper management.

Beware The Dotted Line

Helping out other teams is an important part of increasing your visibility, but beware of the dotted line. The dotted line simply means that you are doing most of your work for someone who isn't your boss. On the organizational chart this is shown with a solid line to your boss and a dotted line to the person you are really working for. Being a dotted line team member is generally precarious, but it is even more dangerous when you're remote.

You need to spend extra time becoming integrated with your team. Creating personal connections with two teams, neither of them your full-time team, is very difficult. Help other teams with smaller projects and try to avoid longer assignments away from your own team.

Team Profile

Subversion - Creating An Open Team

Software is about people much more than it's about code.
- Brian Fitzpatrick

The versions of this book were managed with Subversion.

Subversion is an open source industry-standard tool for centralized version control founded by CollabNet. Subversion has a distributed team and helpful community which would be the envy of any open source project. They have a strong team and a thriving group of contributors. Most of all, the team has become self-sustaining as many of the initial team members have moved on but the team continues to do well. And it almost didn't happen.

Jim Blandy's passion about software development is infectious. He was the first Subversion team member I interviewed and it took me a few days to stop talking about him to everyone I met. The idea for Subversion came while Jim was having lunch with his coworkers from Cygnus Software. They were talking about software and one of them described CVS (a version control system Jim had helped build) as "the worst piece of software I use every day."

Jim talked about this idea with his friend and business partner Karl Fogel and they identified the fundamental problem with CVS: the data wasn't stored properly. Changing the data storage made a new set of commands possible. The new data storage was the fundamental feature of Subversion.

We'll hear more about Michael Pilato in the Open Teams chapter.

Karl and Jim added Brian Fitzpatrick and Ben Collins-Sussman. C. Michael Pilato joined them a little later. This small group created the basics of one the most popular version control systems ever created.

I spoke with Karl a week later and he told me a completely different story. The basic details were the same, but the team dynamics were totally different. Jim told me about a close group of people who had worked together for years and were in almost constant communication. Karl told me about a community of people from all over who came together on IRC and in mailing lists to create a loosely knit distributed team.

The difference was easy to explain. Jim told me about the way Subversion was when it started and Karl told me about the way it was almost eight years later. Jim left the Subversion project after one year and returned to working on RedHat Linux. He only saw the project's expansion and growth from the outside. So how did this team transform itself from a small exclusionary group into a larger international community? It all starts with second class citizens.

Second Class Citizens

Most distributed teams aren't fully distributed. They split their staff between the home office and everywhere else. The group in the home office have an easy time communicating. Walking over to someone's office is simpler than chatting online so the home office group do it more often. After a while they rely on each other more and more. They work together and become productive. All of this feels good, but it excludes the rest of the team.

Second class citizens is the most common problem in distributed teams. And Subversion had it in a big way. The core group was so used to working with each other that they did it without thinking. They had been doing it for years. Everything they did was right —they focused on teamwork, communicated well, and wrote a lot of good code— but it wasn't working.

Subversion started as an open project with a public mailing list, and shared repository, and a public review process. But the core group was too good at working together. They had private email and phone conversations. Notifying the public list became an after thought. Without realizing it they created two teams: core Subversion and everyone else.

Open source projects succeed or fail based on the quality of their community. Jim, Karl, Ben, and Brian together weren't enough to make a product as large and complex as Subversion. They needed help from other people and they weren't getting it.

Making someone feel like a second class citizen will make them leave your project. According to Karl, having two tiers on the team always "becomes visible to the other engineers who are not part of the core. And it doesn't serve any purpose except to increase the core's comfort level with itself."

The core team's interactions with community members was dwindling. The first wave of interested volunteers was leaving and they weren't being replaced by new ones. CollabNet CTO Brian Behlendorf was the one that noticed it first. He saw the team going in the wrong direction before it became a problem.

Faced with the possibility of losing their openness and hurting the project the Subversion team did something amazing. They changed. Changing a team is extraordinarily difficult. First you have to admit to yourself that something is going wrong. Then you have to identify what it is and get everyone to agree on what you should do about it. Taking practices that have worked in the past and abandoning them is one of the most challenging tasks an organization could face.

Transition

Brian Fitzpatrick remembers, "In order to really expand beyond the small group of people that CollabNet was paying to work on [Subversion] we needed to find a way to be inclusive of other people." The Subversion team started their transition by identifying and removing every team practice which contributed to second class citizens. They established a set of rules for the team to make sure it became an open team and stayed that way.

Make All Decisions Publicly

When Jim, Karl, Brian, and Ben talked they made decisions about the project. Often without realizing it. There was just a general feeling that the four of them made a quorum. Ben said, "we thought it was silly to have 'trivial' dialog on the mailing list." They tried it anyway and they immediately "started attracting some unbelievable volunteer developers."

No Phone Calls

> Karl Fogel:
> "All conversation should be held in a public forum by default."

The team continued to have personal conversations, but they don't make decisions about Subversion. Phone calls are a very useful way of communicating with individuals. They aren't very useful for communicating with teams. The core team didn't stop talking with each other; they just stopped making decisions without the rest of the group.

Use the Tools Well

Karl Fogel:
"Use the tools and the metadata in the way that they are meant to be used."

Another part of creating a useful body of information for the team is making sure everything has useful metadata. This particularly applies to using threads well on the Subversion mailing lists.

The Subversion team uses their mailing list as a discussion forum. Patches are submitted, problems are addressed, and features are discussed. From the start they enforced the use of threads for these discussions. Every mailing list post was grouped by subject.

Don't Repeat a Conversation, Link To It

Karl Fogel:
"Don't say something unless you have something new to say."

Making major decisions on the email list created an archive, but the Subversion team wanted to make sure that archive remained useful over time. To make this possible the Subversion team created a strict rule to reference information rather than repeat it. If something had already been discussed by the team they didn't want to repeat the discussion. By consolidating their information the Subversion team created an environment that fostered open collaboration.

No Code Owners

Many teams use an owner or an author comment on source code files to establish who has responsibility for that file. The Subversion team forbids them. Karl explains, "If you put a name at the top it makes other people feel like they have to ask permission before they make changes in that file."

What About Code?

Jim, Karl, Brian, and Ben have all remained friends. I spoke to them separately and they had only good things to say about each other. However, none of these good things had anything to do with code. Not once did any of hem describe their teammate as, *he was a strong coder* or *he was great to work with because of the code he wrote*. All of the praise they had for each other was about how well they communicated.

None of the changes they made to remove the feeling of second class citizens had to do with code. It wasn't a coding problem. The Subversion team almost failed because of a lack of communication and human interaction. Because they didn't have an open team.

The team spent a lot of time and effort focusing on their community. They based that community on four basic principles: politeness, respect, trust, and humility. Members should be polite when they communicate with each other, respectful of other people and their opinions, trusting of the community as a whole, and humble enough to know they don't know everything. Staying focused on these fundamentals allowed them to create a friendly environment where people are happy to work. Subversion still uses the same principles over 10 years later.

The Results

Over time the core team became less insular and included everyone in discussions, decisions, and resolutions of tough technical problems. Today Subversion has 52 blanket committers who are allowed to make changes to any part of the product. There are 71 other project members with more limited focus. The Subversion team also includes 123 people who create, test, build, document, distribute, and translate the Subversion client and server. All of these people are distributed with only very small groups working in the same location.

The Subversion project is very successful. It is a standard for version control in the open source world and has penetrated into corporations where open source has never been before. Over time most of the core team has left the project, but the project is thriving with a new wave of contributors from all around the world. This is the ultimate proof that their efforts to create an open team paid off.

Open Teams

Staying connected to your team is a good start, but it isn't enough. You have to share what you are doing with your team and make them share what they are doing with you. You have to open your team to communication. Creating an open team is always a good idea, but it is essential to succeeding when working remotely.

So what is an open team? Sometimes they are call consensus based teams. Mark Mitchell from Code Sourcery calls them good teams. It is a team where everyone can talk about what they are working on. Have you ever been on a team where there was code you weren't supposed to look at? Code that was owned by a specific person on the team and everyone else was supposed to keep out? That is a big sign of a closed team.

On an open team you can look at everyone's code. That doesn't mean there are no specialities, it just means everyone feels comfortable talking about what they are doing. It also means everyone is happy to ask for help from anyone else on the team.

What If My Team Isn't An Open Team?

Working remotely works much better with an open team because being distributed makes communication more difficult. Your teams needs to go out of its way to make this happen. If you aren't a manager you probably feel like you make your team change the way they work. If you are a manager you know you can't.

If most of your team is located in the same location you probably won't have much luck making them change the way they work just for you. Calling a meeting and announcing that you have read a new book and want to make the entire team change will get you nowhere. However, you have an awesome power to change your team; even more than your manager. Your manager can only tell the team to change, you can show them how.

When you want to open your team the process is simple: fake it until you make it. Act like your team is open and it will become open. Open yourself up to your team's criticism. It is simple, but also really scary.

Ask someone on your team for help or advice. Do it publicly, and make it someone more junior on the team than you are. Start by asking for a code review. Make it clear you want their help to make your code better. Then comes the hardest part: take some of their advice and thank them publicly.

Let everyone on your team see what happened. You picked someone, asked for help, make your code better, and thanked them. Then repeat the process again. It doesn't take long before one of your coworkers will try it, probably the person you were asking for help. This process will make your code better, but it

Code Kingdoms

A code kingdom occurs when a small number of people, usually just one, want to own an area of the code and keep everyone else out. The king or queen of that code will work hard to make sure they maintain complete control over the code and strongly discourage anyone else from looking at it.

The Subversion team tracks code kingdoms with a metric they call the "bus factor." It tracks how many people need to be hit by a bus before all knowledge of a feature or subsystem is lost. The higher your bus factor the more redundancy you have and the closer you will be to having an open team.

will also make your team open up.

Bragging

Bragging is another way to open up your team, but only if you do it right. Brag inclusively rather than exclusively. Don't make it a competition.

Making bragging an open forum for the rest of the team and don't make it competitive. The message should be "let's all show the great work we have done."

You have the power to show your teammates a different way to work that will benefit you and them.

Micro Demos

Micro demos are one of my favorite ways to show my team what I'm working on. The key to a good micro demo is speed. The best micro demos are about five minutes long. Much longer than that and people will feel like it is a disruption.

That gives you a very short time to introduce what you are showing them, give them demonstration, and make them care. Show your team quick little snippets of what you are feeling really good about. Did you find a better way to solve a problem or a improve the usability of a feature? Are you really proud of the code you just wrote? Show your team.

The Remote Presentations chapter will show you even more about giving presentations and demos.

Open Source and Open Teams

Many open sources teams are accomplished at being open. They have to be. Many open source teams are made up of volunteers who don't know each other. A lot of benefit to working on an open source project is getting to work in an open source team. Your code gets stronger, you make better projects, and you get a better working environment.

The Social Currency of Open Source

"The best way to be influential is to be influenced."
-Brian Fitzpatrick

Every team of people runs on the exchange of social currency. Every team member wants to increase their social currency and become a more senior member of the team. If you ask someone for help and then use their help well, you will move up in the social structure.

Gaining respect by asking for help is the opposite of most corporate teams. Those teams often see needing help as a sign of weakness. You won't find this process spelled out in a company handbook, but it is easy to test for. In your team, when you ask for help do you feel like the person you asked is doing you a favor? Are you reluctant to ask them because you are afraid of what other people will think? If you are, then the flow of social currency is clear: give someone help and you are rewarded, ask for help and you are penalized.

Open source teams don't have a monopoly on positive culture and some corporate teams are run in an open way. Open source is a useful way to talk about these two styles of teams. Mark further reinforces the idea of open teams when he repeats something he tells every member of his team.

> Mark Mitchell:
> "There's no shame in getting stuck, there's no shame in not knowing, and there's no prize for figuring out something that someone else has already figured out. The point is to work as a team and ask for help and collaborate with each other; to learn from each other and move the overall thing forward."

There is nothing wrong with being rewarded for helping someone. But when you are penalized for asking for help, you have the beginnings of a territorial and competitive team. Being remote will increase this territoriality and it leaves you at a disadvantage. Being in the office gives you personal access to people. It is a lot easier to be pushy when you are in front of someone and very easy to be ignored when you are remote. The only way to win this game is to change the rules.

Understanding Open Teams

You are personally responsible for making your team an open team. And so are all of your teammates. You create an open team by redefining success. The success of a product is simple: it sells. The success of an engineer is more complicated. Writing code, creating features, and fixing bugs are not enough to make you successful. You won't be a successful engineer unless your team is successful too. Your most powerful tools for helping your team succeed are knowledge sharing and peer review.

Peer Review In Action

An open team can be difficult to define. Being open is an ongoing process rather than a single event. It is a team-based way of thinking about your work. Let's look at an example:

The Subversion team is a textbook case of a successful open team. They organize a large group of people and keep them involved with all aspects of their product. They strongly discourage code kingdoms and don't allow any owner or author information in their source code.

Subversion is an open source team and all of their team communication is publicly available. Here is are portions of a real mailing list thread which took place between June 20, 2003 and July 30, 2003 on the Subversion developer mailing list. The thread started with a single email and the simple subject "svn commit performance."

The full thread is available in the Subversion Mailing List Archive.

On June 20, 2003 Chia-liang Kao brought up an issue with the Subversion team. He had observed some performance problems with the commit command. This is the command which commits changes from a client machine to the Subversion server.

Like many performance problems, this one only happened under specific conditions. The first step Chia-liang took was to introduce the topic and give a decent amount of background information. The Subversion team tries to make raising an issue as simple as possible. In this case Chia-liang sent an email.

His email to the Subversion developer mailing list which said, "Recently I noticed the svn commit performs poorly before bringing up the editor." He described where he was seeing the problem (during an import from the directory - svn co) and how to reproduce it. He also made a few guesses as to the cause of the problem:

Subversion Terms Quick Guide

co	checkout or import
commit	check-in
editor	the place where you edit your comments
diff	show differences
file:///	the path to the local server
full committer	someone with permission to change the Subversion product
svn	the Subversion command line

> Chia-liang Kao:
> If i do a "svn co file:///repo/trunk svn.co", edit a file in a deep directory, "svn diff" gives me the result immediately, while it takes 3 or 4 seconds for "svn commit" to bring up the editor. this is still acceptable.
>
> But if i do "svn co file:///repo svn.co.full", edit the same file in under trunk/'s deep direcotry. "svn diff trunk" still gives me the result immediately. but now "svn commit trunk" takes about 7 seconds to fire the editor. manually applying 6301:6302,6312:6313 for 0.24.2 takes 10 seconds even.
> It seems the number of tags made the difference, since i didn't notice this lag when i had no tags. but i think "svn commit trunk" should be irrevelent to tags/ ?

Chia-liang didn't know the solution yet, or even if the problem was worth solving, he just saw a problem and brought it up with the team. He made sure to give as much information as possible so other people could follow what he was saying. He also wrote in a clear and easy-to-understand manner. The only problem I could find is that he didn't use proper capitalization. Soon after he sent this email, he sent a second email with a theory of what could be causing the problem.

His first email did a good job representing the severity of the problem. Chia-liang doesn't try to make this issue more important than it really is. He simply describes the behavior and how it will affect the user. He gives an objective assessment when he reports that the commit will take up to 10 seconds.

The first responder was C. Michael Pilato. Mike adds to the theory from Chia-liang's second email. He also acknowledges Chia-liang's question as one worth investigating by taking the time to understand his problem and respond to it. This is also implicitly validating Chia-liang's work. By understanding and helping to resolve the issue Mike is showing that he thinks the issue was well-researched and is worthy of more of his time.

> C. Michael Pilato:

And I'm gonna guess that what you saw was 'lock' files being written under tags/*, yes?

For reasons that I honestly don't understand, 'svn commit foo' does a full recursive lock on foo's parent directory. I've noticed this behavior for some time now, but haven't had the chance to investigate.

By giving a reasoned and thoughtful response Mike has made it clear that he thinks this is a worthwhile topic for the team and encouraged Chia-liang to continue to investigate. Mike and Chia-liang discuss this issue throughout the day and by the end of the day Mike is working on a patch. A few more people enter the conversation and it turns out the problem is a little more difficult than the team originally thought. Mike and Chia-liang work on the patch for about three weeks with some help from other team members. On July 10, 2003 Chia-liang proposes a patch.

Chia-liang was a full committer and had been an active member of the project for a long time. He was a senior member of the team, but he didn't just make the patch. His first step was to make the patch available for the team to review so he could get feedback. And the feedback that he got was more than just a quick code review.

Chia-liang received many comments on his patch. A particularly thorough comment came from Philip Martin.

Phillip Martin:
No, don't use the probe functions, just use the information in the entries, call `apr_hash_get` to see if access batons exist, and call `do_open` to open any missing access batons.

Look at my original algorithm again:

```
Get P the path for adm_access
Get R the relative path of descendant and P
Split R into components.
For each component C in R
Get the access baton for P
If no access baton then error
Get the entry for C in P
If no entry for C then error
If C is not a directory return
Construct new P = P/C
Get access baton for P
If no access baton
Open new access baton for P
If that fails then error
```

Note the step "If C is not a directory return" this will allow the descendant path to be a file. At this point, before the algorithm returns, it should check that C is the last component of R, if not it should return an error.

This comment is the epitome of good peer review. First, it is very specific. It avoids general comments and uses a laser-like focus to precisely address issues in the code. Second, he proposes a solution. It is easy to say, “this is wrong,” but it is much more helpful to give a suggestion for how to fix it. Third, this comment doesn’t have any blame or judgement. It simply presents the facts in a straightforward manner and lets them stand on their own. It doesn’t come across as a putdown or attack on Chia-liang’s abilities in any way. Finally, it is clearly written and easily understood. Even if you don’t know anything about the code in question you can still follow most of what Philip is saying. This superb example of team communication illustrates why the Subversion project has been so successful.

It’s not surprising that Chia-liang took these comments and made the suggested changes. Responding to a specific issue with large amounts of clear objective data almost always gets listened to. The team went back and forth a couple more times and then made the patch.

This patch succeeded for a number of reasons. Chia-liang Kao brought up the problem in an open-ended way. He did not try to force a solution; he just suggested one and then took feedback from the rest of his team. He also brought up the issue with the whole team rather than picking specific people. Not everyone responded, but it gave everyone a chance to be included if they wanted to be. When C. Michael Pilato made comments he stayed positive. He did a good job of expressing his concerns without becoming judgemental. When Philip Martin reviewed the patch he gave specific suggestions. He also used a straightforward nonjudgmental style. Addressing specific issues with constructive criticism is an implicit way of being supportive.

This issue took a little over a month to report, analyze, and fix. It involved six different people from the Subversion team and it was discussed at length. You might look at this example and wonder how the team ever gets anything finished if it takes six weeks and most of the core team to make a patch. In fact, the Subversion team has a long history of being an extraordinarily productive one. The explanation of this incongruity is one of the most unobvious aspects of peer review: helping other people will make you more productive.

Peer Review Makes More Time

In most teams when you help someone else you get the credit for solving their problems. You took time away from your work to help them. That means that you have less time to finish your own work. In essence, the time you spent helping that other person was a direct cost to you. The logic is simple: you had X amount of time, you gave Y amount of that time away, so you have X - Y amount of time left. This equation is straightforward, logical, and completely inapplicable to software development.

Spending your time helping other people leaves more time for your work. This sounds impossible, but it is true for your team. When you add to the general knowledge of your team you are helping yourself. Working in a smarter, more capable team will always help you. When you work together you can get more done than when you work separately.

We can quantify this theory with a simple example. Assume you are working on a new subsystem. If left alone, you would spend 1 week designing the subsystem, 2 weeks implementing the subsystem, 1 week refining the subsystem, and 3 weeks fixing bugs. The total time would be 7 weeks.

You can create better software faster when you have more people looking at it. This doesn't mean your team reads every line of your code. It means that you spend time talking with them about the decisions you are making and why.

If you completed the same process in a peer-reviewed team it might look like this: 1 week of overhead helping out other people, 1 week designing the subsystem, 2 weeks implementing and reviewing the subsystem, and 2 weeks fixing bugs. The total time would be 6 weeks. You save a week even though you spent a week helping other people. The intensive review of your ideas will lead to a faster turnaround with fewer bugs. It also decreases the risk that you will have to redesign the system because you didn't understand one of the requirements.

Peer review will make your code more readable and more stable with fewer bugs. The increased communication of the open team model will also make it much easier for you to stay connected with your team when you are remote.

Small Changes Make A Big Change

The next section of this book will tell you how to start building an open team. It focuses on peer review and code reviews. Chances are that these concepts aren't new to you. You have probably heard them for years. You probably find the idea that you alone can revolutionize your team by suggesting code reviews a little tough to believe. You aren't the only one. Many people have trouble with this concept.

Be Patient

Creating an open team based will make a big difference in the amount you can achieve and the success of your team, but it doesn't happen overnight. When you first begin this process it will be disruptive and take more time than it saves. Like any new process, it takes a little while for your team to adapt to it.

The truth is that you can't do it by yourself. You can only start the process. The rest of your team needs to take it forward. Creating an open team isn't about the code review or the peer review meeting. An open team happens in the time between the meetings. It happens with a set of phone calls, IM or IRC conversations, and documents that share the knowledge and responsibility of creating a product with the entire team. It is a subtle change that affects the way you work every day.

The other truth is that this issue isn't black and white. Creating an open team is a constant effort because you can never be a completely open team. Your team will always be open in some ways and closed in others. You should always push to be more open because your team is always in danger of becoming more closed.

Real Life Isn't Like The Movies

In a movie the action will build to a climax where the hero stands up for themselves and, with a single defiant act, changes their life forever. The protagonist delivers that final speech, closes the major deal, or resorts to violence --you've seen those movies where someone punches out their boss-- and nothing is ever the same. Real life doesn't work like that.

If you make a big speech to your team about why they need to be an open team they will think you are spouting something you read in a book. They will probably ignore you. You help your team be more open in little ways every day. Pick up the phone and ask for someone's opinion. Take a little extra time to let someone know what you are working on. It won't take long for you to see that you can be better at your work with a little help.

Asking for help, and using it well, will quickly make you one of the most productive members of your team. Seeing how productive you can be will convince your team members to try peer review faster than any speech. You have the power to be an example. Show your team that helping each other can be a reality rather than a utopian vision.

Changing your world with peer review may seem idealistic, but it really can work. Team culture is malleable, and you can shape it into something better for your entire team by putting the open team philosophy into action.

Put It Into Action

Now you have an understanding of the theory of peer review and open teams, it's time to put it into action. You have the power to change your team by example. Pick a project you are working on and try to do it in a peer-reviewed way. Start with a one-on-one conversation. Choose someone you feel comfortable with, call them up, and talk about your work. Tell them about the decisions you are making and why you are making them. Try to give them a good understanding of your system and why it is the way it is. Don't just talk about what is going well; let them know where you are having problems.

The next step is more difficult: take that other person's advice. You don't need to do everything they say, but choose some change they suggested and implement it. The purpose of talking about your ideas is not to find out how great they are, but how you can improve them. Once you have gone through this process a few times with just one person you can try a peer review meeting with more people from your team. As your team becomes more experienced you can hold peer reviews with technologies that do not happen in real time.

How To Succeed In Your First Peer Review

Stay positive. You will get a lot of feedback and most of it will be negative. It takes many people a while to learn how to give useful feedback. Stay positive. Remember, this is a learning process for you and for the rest of your team.

Prepare the project, don't prepare for the meeting. A peer review meeting is not a presentation. The way to get ready for it meeting is by improving your existing code and documentation. It isn't by creating a complex presentation. Just focus on improving the features and the code itself.

Keep the group small. The first peer review meeting should be fewer than six people including you. The group can get larger later.

Watch out for groupthink. The loudest person will probably speak first. Don't let them dominate the conversation. Solicit feedback from everyone in the group.

Don't try to impress people with complexity. Choose something simple for your first peer review. Don't try to impress people with something complicated; it will only make the peer review more difficult.

Start with an interface. It is much easier to review an interface rather than an implementation. Jim Blandy from the Subversion and Mozilla teams poses a simple questions, "Talking about interfaces is a proxy for talking about design. Is that something that [you] could imagine implementing?"

Once you have finished your first peer review meeting you must incorporate some of the feedback you got. Then call a second meeting. Get the process started and make it clear that this wasn't a one-time event.

Code Style

Follow whatever style is your team uses. Your team may or may not enforce a specific coding style, but it is still a good idea to make your code look like everyone else's. It doesn't really matter which line you put your curly braces on so you might as well make your code fit with the rest of the team. This will help other people read your code and make it easier to integrate and share with the rest of your team.

Peer Review As A State of Mind

Peer review meetings are the focal point of the peer review process, but they are only a small part. Peer review is a culture tool affects the way you work every day. The biggest difference is that you don't just talk about what you are doing or how you are doing it; you talk about why you are doing it. Why you are making a change is more important than the change you are making. The code you write today may be replaced tomorrow, but the reason for changing it has a little more permanence.

Document Why, Not Just How or What

Documentation is vital to the completeness of any coding project. If your code doesn't have documentation, then it isn't done. Most coders are good at documenting how things work. You will see a lot of comments like, "added a new method to the API" and "iterate over these values to find the one we are looking for." These comments are useful because they make the code more readable, but they don't tell the whole story.

Don't restrict your documentation to code comments. You have many opportunities to talk about your code like check-in comments, bug comments, and design documents. Take advantage of them.

Use the tools well

Karl Fogel:

"Use the tools and the metadata in the way that they are meant to be used."

Another part of creating a useful repository of information is making sure it has useful metadata. This particularly applies to using threads well on the Subversion mailing lists.

The Subversion team uses their mailing list as a discussion forum. Patches are submitted, problems are addressed, and features are discussed. From the start they enforced the use of threads for these discussions. Every mailing list post was grouped by subject.

Good and bad comments

Check-in comments are a good representation of what you do every day. Good check-in comments will give you and your team the information you need to understand the changes you are making. Bad check-in comments will make your version control history useless. Productivity for engineers is often measured by the code they produce, but that productivity can be difficult to track. The number of lines written or number of functions implemented are not useful metrics. Looking at check-in comments is an easy way to track the progress of individuals when they are writing code.

The first step is to ensure that every time someone checks into the version control system they provide a comment. Many organizations don't enforce check-in comments and many engineers forget to add them. Every single check-in you make should have a comment. It doesn't need to be paragraphs long, just enough to get a quick idea of what changed and why you changed it.

A good check-in comment will give the reader a quick understanding of the change that was made.

Check-in comments should:

1. Give a simple and straightforward explanation of the change.
2. Be written in complete sentences.
3. Talk about why you made the change and not just what you changed.
4. Mention other people who helped with the change.
5. List the numbers for the relevant bugs.

Check-in comments should not:

1. Be empty.
2. List just a bug number.
3. Give formatting descriptions like: "changed two lines"
4. Make sense only to you.
5. Document every single change made to the file.

Code Reviews

Code review is another fundamental part of being an open team. You should make your code available for code review. Not because the reviewer will find bugs, but because it will make you write better code.

The code you write is the final output of your development process. It shouldn't be left out of the knowledge sharing of your open team. Some teams have specific policies for when and how code should be reviewed. Others are more relaxed about it. Either is fine as long as the code review remains positive and focuses on specific ways to make improvements.

How To Get A Good Code Review

Getting a good code review is all about preparation. Before you ask someone for a code review your code should be:

- Compiled
- Tested
- Formatted
- Documented
- Cleaned up so it's easy to read

Getting a good code review is also about accepting the reviewer's advice, so don't get defensive. You are asking them for help and you have to be ready to take it. The purpose of the code review is to make the code better, not to get someone else to tell you that it was perfect already. You are there to talk about why you wrote the code the way you did and get feedback. When you are done with the code review, the other person should feel comfortable talking about your change and explaining it to someone else.

Accepting criticisms from other people can be difficult. It has a lot to do with how you are feeling and how receptive you are to hearing what the other person has to say. Mike Pilato from the Subversion team has called the code reviews he received "invaluable," but he also admits:

Michael Pilato:
"Some days you may wake up and you're cranky and you think, 'I know the minute I send this patch off, here comes that code review and I'm gonna get smacked around for it.' A lot of how you receive code review is all about your own attitude."

Tag Team Comments

The most thorough form of code review is probably tag team comments. I have only achieved this level of collaboration once in my career. Tag team comments are simple; have someone else comment your code while you add comments to theirs. This has many of the benefits of extreme programming without having to share a keyboard.

Adding comments to someone else's code forces you to really understand what they are doing rather than just giving the code a cursory glance. It also leads to much better code comments. Tag team comments might be a little too extreme for you, but you should always encourage other people to add comments to your code when they are reading through it or fixing bugs.

How To Give A Good Code Review

Stay positive. How you say something matters as much as what you say and giving a good code review is all about tone. When someone else presents their work to you they are in a vulnerable position. Try to make them feel a little more at ease. Ask questions instead of making statements. If something doesn't look right, ask them why they did it that way instead of telling them they did it wrong.

Focus on content, not style. There is more than one way to solve every problem. If someone else's solution is different than yours, take it as an opportunity to learn something new. If your team has code style standards you should help enforce them, but don't penalize someone for solving a problem in a different way than you would have.

Don't debug during the code review. When someone brings you code to be reviewed that code should already have been compiled and tested. A code review can be helpful in finding bugs, but it is primarily a place to discuss the implementation. Stay focused on understanding why the person made the changes they made and don't try to be a human debugger.

All Team Members Should Review Code

It's common to have a senior team member or an architect oversee the product and perform all the code reviews. This is a bad idea. The obvious reason is that having only one person perform code reviews will be a bottleneck to productivity for the team. The more subtle issues involve learning and shared responsibility.

When you give someone a code review you share a little responsibility for their code. You are looking at it and understanding it. You are also teaching each other how to be better programmers. No single team member has a monopoly on wisdom. Sharing code review responsibilities is a way to share knowledge and help everyone become a better engineer.

Online Code Review Tools

When you do a remote code review with one person it is easiest to do it in real time. Just pick up the phone and talk about the code while both parties are looking at it. However, time constraints can make it advantageous to do a code review asynchronously. Performing code review via email is very common in the open source community. There is no reason a code review must be done in real time.

There is also a growing number of online code review tools. These tools will show you differences, let you make comments, and generally help you organize the information for your code review. Some of these systems run on servers which your company must install and other run on third party servers. If you are using a tool hosted on a third-party server, remember that your code will be out of your control and possibly available to the public.

What About The Managers?

So far this sounds great. The team is communicating well. Everyone is sharing responsibility for the entire project and all the teams members are learning from each other and writing better code. So... What about the managers?

You may find it difficult to work with your manager when you work remotely. The answer is don't. Don't work with your manager like a manager. Work with them like a team member. Encourage them to be part of the team rather than above it.

Your manager is just another member of your team. They need things from you and you need things from them. Ask them for a code review. Your manager doesn't know anything about code? Ask them anyway. Show them what you are doing. Include them in technical discussions. Don't treat your manager like an outsider.

Making your team open will make it very easy to manage. Everything that is happening on the team is clear and your manager never needs to wonder what you are up to.

My Manager Can't Code

It is often the case that the best technical managers are also programmers, but not always. If your manager isn't technical it can still work well. One of my favorite managers never wrote a single line of code for our team.

Instead of writing code he tested and designed the product. He was our first user. He tried every new feature and gave useful feedback. He tested the product and wrote a lot of good bugs.

Your manager can be a very useful team member without coding. Tell them you think so. Make it clear that they can help and you want them to. Invite them by showing what you are working on and getting their feedback. Make it clear you're looking for their opinion, not their approval. Invite your manager to all design meetings and encourage them to speak up.

It may sound funny, but some managers are a little scared of their team. This is especially true of non-technical managers. They are surrounded by a group of people with skills they don't have. Make it clear they don't need to program to be a very productive member of the team.

Distributed Open Teams

This chapter has mentioned very little about being telecommuting. Creating an open team is good for all teams, not just distributed ones. Openness is especially important when the team is distributed. Distributed teams have a stronger tendency to become separated. The isolation of team members will lead to territoriality, a difficult-to-maintain product, and a very unpleasant working environment.

You are also much less likely to get promoted. Open teams make it much easier for remote engineers (and everyone else) to have their contributions recognized. This benefits all engineers on the team and makes it a more even playing field for the remote ones.

Open teams share knowledge much more efficiently than closed ones because they get a lot more practice. Everyone on an open team will know how to contact everyone else and be familiar with their communication styles. This makes it possible to communicate in a simple email instead of a long conversation. Lowering the time and effort it takes to communicate with your team is your ultimate goal as a remote engineer.

You just can't afford to allow your team to stay closed. You must push for a more open team with full communication and peer review, or you will be left out. If your team doesn't communicate well then it will be very difficult to be part of the team.

SECTION 3

OPEN COMMUNICATION

Team Communication Fundamentals

There are some fundamental questions you need to answer in order to be a productive member of a software team. You need to answer these questions every day and the answers are always slowly changing. The next two chapters focus on the basics of communicating in real-time and written communication. Throughout both of these chapter we will be returning back to these fundamental questions and how each communication mechanism helps you answer them.

How can I ask simple questions?

What time is the meeting? Where is the string utility function? There needs to be a simple way to ask questions like this. Should you send an email, ask in IRC, or just shout over the cubicle wall?

What should I be working on?

At any given time you need to know what you should be doing. Sometimes this comes from a project tracking system and other times you just know. What is everyone else working on? Knowing what the rest of your team is doing isn't just for managers, it is a fundamental part of the job of every engineer.

How can I ask questions without being disruptive?

You have to feel comfortable asking questions without worrying that you are preventing other team members from getting their work done.

Where is the information I need?

When you are looking at an unfamiliar part of your product or a new tool you need a method for finding information about how it works. Is there a design document, code comments, or should you just look at the code? The answer is different for every situation.

Communicating In Real-Time

Mozilla has about 200 employees spread around the world. They work from big campuses like the one in Mountain View, California, small offices in places like Beijing, and out of their homes in faraway locations like Dunedin, New Zealand and Singapore. However, most of the people working on Mozilla products don't work for Mozilla. At the last conference for Mozilla developers there were two volunteers for every Mozilla employee.

At Mozilla there is an overwhelming amount of communication options. Project members have the choice of email lists, newsgroups, blogs, wiki pages, and more, but more than anything else the company runs on IRC. IRC helps all of the developers stay connected with each other, their community, and their customers.

Internet Relay Chat

When Daniel Einspanjer, a metrics software engineer at Mozilla, wants to see what the rest of his team is doing, he turns to Internet Relay Chat (IRC). Instant messenger is more focused on one-on-one conversation while IRC specializes in group of chatting. Everyone is in the same space chatting together. Daniel works from his home in Boston, Massachusetts and IRC lets him stay connected with his team.

IRC also supports smaller rooms that are formed for members of specific teams, people interested in the same topic, or just a group of friends. Daniel told me that IRC works like the company water cooler; a place to chat with friends and catch up with coworkers.

If the IRC chat rooms at Mozilla are a cocktail party, then the chat rooms at MediaWiki are a crowded sweaty rave filled with loud music, angry party goers, and a big guy who just barfed on your shoes. MediaWiki makes the software that runs Wikipedia, and the high-profile nature of that project makes sure their chat room is always crowded.

Over the three weeks I watched the MediaWiki IRC I rarely saw attendance fall below 175 people. The conversation never stopped, there were always multiple topics, and many of the posts were just mean. It left me wondering how IRC could be anything more than a time-consuming nuisance.

The IRC at Mozilla is the total opposite of the cacophony of the one at MediaWiki. Mozilla IRC rooms rarely have more than 25 people. Often it is less than 10. And most of the time they are quiet. Most of the activity comes when people are starting or ending their days; talking about problems, bragging about solutions, and catching up with each other.

IRC is a tool, and like all tools it can be used well or used poorly. When IRC works it connects far-flung teams; when it doesn't it is just noise.

The Basics

If you are new to IRC, here are a few basics. You need an IRC client. This is the program that knows how to connect you with an IRC server. The IRC server is a place where many people come to talk. Once you get to the server you need to join a room. Join the room with `/join #some_room`. Once you are in a room you are ready to start.

Minimize the IRC window when you are not using it. An IRC chat room is a shared space for your entire team. People will be talking to each other without needing your input. Close or minimize the IRC window to get some work done. The software will alert you if someone says your name or uses any of your predefined stalk words.

Chatzilla

Chatzilla is a full-featured IRC client written on top of the Mozilla framework. You can install Chatzilla automatically as part of your Mozilla browser installation. It's also free.

Call people by username. If you want to talk to someone specific you need to reference them by username. That will cause the chat software to alert that person that you need them. For example, "Hey Zack, are we meeting this afternoon?"

Show your status by changing your username. It is a common practice to change your username to show your availability. Normally I am just Zack, but I can also be Zack_busy, Zack_inmeeting, and Zack_lunch.

Separate different work groups into different rooms. IRC supports the concept of rooms or channels. Rooms are a way to separate conversations and make them more manageable. Keep the rooms small. Even five or six active participants can overwhelm a room. A common pattern is to have a general room where you can get somebody's attention and then private rooms where you can hold smaller conversations.

Keep it short, or pick up the phone. When you ask questions in IRC make them short and to the point. Don't ask longer rambling questions and don't engage in longer conversations. Tying up the IRC channel with a long conversation will make it difficult for your coworkers to ask their simple questions. If the conversation gets too long then you should switch to a conference call.

IRC robots

XULRunner is the platform that popular Mozilla programs like Firefox the Thunderbird run on.

When Benjamin Smedberg, the coordinator and lead developer of the Mozilla XULRunner project, breaks the build a robot tells him so. Benjamin works from his home in Pennsylvania, far away from the Mozilla build servers in Mountain View. But if something goes wrong with his project Benjamin knows about it almost instantly. There is an IRC robot (a computer program) that watches the product build and tells him, and everyone else in the IRC room, when it breaks.

The robot can't always tell who broke the build. It could have been Benjamin, but it also could be someone else. More importantly, the robot can't tell who the best person to fix the problem is. By using IRC the robot can stick its head up and say, "Hey guys. The build is broken. Anyone know how to fix it?"

Stalk Words

Most IRC clients support a feature called stalk words. This feature lets you define words other than your name that you want to be alerted to. You could have the program alert you any time someone says *build* and *broken* so you know to fix it. You could also be alerted when someone mentions the subsystem you work on. Stalk words let you ignore the IRC window without worrying that you are missing something important.

This robot could also send Benjamin's team an email, but IRC has two big advantages. First, the robot can talk to the people who are online right now. They are the people who are most likely to fix the build fast. Second, people won't have email telling them the build is broken when the build has been fixed hours ago.

IRC is also a welcoming place for robots. The simple chat protocol makes it very easy to write computer programs that work with IRC, and there are a lot of them. IRC robots can tell you:

- When the build is broken
- If a new bug is added
- If an old bugs gets fixed
- When someone checks in code
- When integration tests fail

The team could also use email, but IRC is much less disruptive. Many companies don't want email notifications every time new code is checked in because the volume of email can be disruptive. With an IRC channel there is no disruption of your work. Log into it when you care and turn it off when you don't.

IRC robots don't have to talk; they can also just listen. Add an IRC robot to archive all of your conversations so you can refer back to them later.

Why you should try IRC

IRC is much more inclusive than IM or email for one simple reason: you don't have to know someone's name to talk to them. When I want to email someone I need to know their email address. It is the same for IM. On IRC I just log in and see who is there. When I was researching this book Daniel Einspanjer told me a lot about how Mozilla works. He also helped me talk to other Mozilla team members. I didn't know Daniel ahead of time. I met him through IRC.

My meeting Daniel is a sort of macro example, but it works on a micro scale too. Have you ever had a question and not known who to ask? Ever a had a team issue and didn't know who to tell about it? IRC lets you reach out to any team member who is online.

IRC also goes a long way towards preventing exclusionary conversations. When you talk to one person in an instant message you are sharing your ideas with them. Convince them that you are right and the two of you have made a decision. However, the rest of your team is out of the loop. They don't know what decision you made or why. Do the same thing over IRC and other people can get involved. Instead of making a decision with just two people you can involve the rest of the team.

Working remotely is all about creating connections with your team. IRC lets you create those connections in a way that scales for your entire team. And if you needed another reason to use IRC: it's free!

irc.freenode.net

Freenode is an IRC server open to the public and available for open source and community based projects. Many open source projects maintain rooms on this server and it can be a good place to start IRC with your team.

Instant Messaging

When instant messenger first came to my workplace I totally blew it. In my defense, I had already had two bad experiences with IM. The first one came when I was visiting an old girlfriend at college. Her computer was in her dorm room and we were interrupted with about 50 separate IM chats a day.

My second bad experience was still happening when my coworkers started using IM. A group of my friends had convinced me to sign up with Yahoo! instant messenger and become their buddy. They were mostly still in college and didn't really understand what it was like to have a full-time job. They IMed me all day long with messages like "what are you doing?"

I know it isn't an excuse, but it helps explain why I flat out refused to use IM with my coworkers. That lasted for a couple of weeks before I caved. Within a few days I started seeing the power of the tool and became and abusive IM user. I demanded answers to my questions immediately often walked over to someone's cube to complain if they didn't answer me fast enough. Yes, we were using IM even though we were all in the same office.

A lot of years have passed since I first started using IM and I've come to a better place with it. I'm more patient and I use it sparingly. I've also come to realize what a disruptive technology IM can be. There are times when I wouldn't want to disrupt someone with a phone call that IM feels OK to me.

The way to make IM a useful tool instead of a constant interruption is to use it sparingly. Keep your chats short and don't bother people for information you could easily find on your own. The next sections present some examples which will help you streamline the way you use IM and help you stay connected with your team.

Avoid long conversations

Short questions with short answers are the sweet spot of IM: "What time is the meeting?" or "Do you have a utility for parsing URLs?" "How are we going to organize our new team? " will work better over the phone.

IM is not the place to discuss controversial or open-ended issues. If you want to ask questions that start with "I know you don't agree with this, but..." or "have you thought about..." then use email or the telephone.

Frame the question

When you ask someone a question it is fresh in your mind. You know what you want to say and you have (hopefully) been thinking about it for a little while before you ask the question. This can lead you to assume the other person knows what you are talking about.

I have received many instant messages that start with, "I'm curious what you think about that bug." I've sent a few like that too. It would be much better to say, "I'm looking at bug 256 in the order processing code. Do you have any idea what might be wrong?" The second example let's the other person give you useful feedback immediately instead of wating time asking for more information.

IM conversations often contain long pauses while someone completes requested work or researches an issue. During that time the other party will have moved on to other tasks. It is important to frame your questions and responses.

If you haven't sent that person a message in over 10 minutes then you should assume you are having a new conversation.

Know when the conversation should end

Even a well-planned chat about a simple issue can devolve into a long drawn-out conversation. It is important to reevaluate your conversation frequently and see if it should be switched to the telephone or wrapped up.

Communicate well about time

If you send someone an IM they will probably wait for you to finish and when they ask you a question they will probably wait for the answer. A few minutes while you look something up may seem short to you, but they will be long to the other person. Before you stop typing for a little while let the other person know if they should wait for you or not.

Avoid unwieldy acronyms and abbreviations

> Zack: Hello Justin. That customer meeting was very interesting.
> Justin: w/e i was ttly zzz u2? ;>)

Text abbreviations are common in IM and can make the message faster to type, but don't go too far. Most organizations will use a small set of abbreviations. If you are unsure, then take the extra few seconds to type the whole word. Your coworkers will appreciate not having to decode your messages.

Choosing A Username

When you join a new IM or IRC server you will have the chance to choose a username. Choose a username which clearly identifies you. I normally use zgrossbart. Who you are should be clear to someone who doesn't know you.

Status

You have to let your team know what you are up to and if you are available. IM is a good real-time tool for showing status. IM clients include the ability to set personal status and see the status of other members of your team. If you are busy and don't want to be interrupted you can use your IM status to indicate that. Use your status messages to let your team know when they can expect your response to their questions.

Security

Whoever hosts your IM account has total access to every message you send. When you send an IM through Yahoo! they can see every word. If you are worried about security you need to suggest that your company install an internal IM server which can be secured and logged for just the members of your company.

Conference Calls

Jono DiCarlo and Atul Varma work in a small group within Mozilla called Labs where they focus on researching new directions for Mozilla. One of their new projects, Ubiquity, helps you create mashups of different websites. The system understands English and lets people with very little technical ability create complex web page combinations.

Natural language parsing is very difficult and Jono and Atul get a lot of help from experts in the field. They hold conference calls with natural language parsing experts, as well as others, about once a week. Jono and Atul work in the same office in Mountain View, but they dial into the call from separate offices so everyone on the call is equally represented.

Jono and Atul are the leaders of the call, but they don't want to exclude anyone else. Being in the same room would make it easier to talk with each other than the other people on the phone. Over time they would start to dominate the conversation. Instead they treat everyone equally so that everyone can be heard.

A well-run conference call doesn't feel like one at all. It feels like a group of people sharing ideas. Nothing however, will make you feel more removed from your team than a poorly run conference call. Some people dial in late, others can't be heard, and everyone shouts because they are afraid they won't get a turn to speak. It doesn't have to be like that. In a well-run conference call, you never have to shout to make your voice heard.

Plan ahead

The first step to planning a good conference call is making people feel calm. The more calm and comfortable people are, the more likely they are to work well together. Jono and Atul's conference calls work so well because they help people feel comfortable.

Let people know what to expect. You don't need a full agenda. Just let people know what to expect with a short email ahead of time.

Notify everyone who should attend. It seems really simple, but a lot of people forget to invite everyone.

Choose a place for your conference call. If you are planning a meeting which includes a group of people in the same location then you should book a conference room for them.

Plan at least one day in advance. Last-minute meetings will happen, but they shouldn't be the norm. Give people enough time to plan their day around your meeting and to get back to you if they need you to change the time or place.

Tell people how to connect to the call. With a small group of people you can call them directly. Include your number in the call invitation and make it clear if you will call them or they should call you. For larger conference calls you should use a third-party conference call service. Include the number of the service and participant passcode in the meeting invitation.

Create a conference call cheat sheet

When you are part of a fully distributed team you share the responsibilities of remote communication with the your coworkers. If everyone needs to dial in then you are all focused on making it work. If you are the only remote person on the team then you have a larger part of the responsibility to make remote communication work.

Help your team with a conference call cheat sheet. Post the cheat sheet in your team's regular conference room. If you can't do it personally then ask a friend. Help your team make conference calls go smoothly by leaving the information they need in an easy-to-find place.

Your cheat sheet should include:

There is a free conference call cheat sheet template available at The One Minute Commute blog.

- Outline of preset meeting times
- The conference call number
- The conference call pass code
- Your contact information: Name, Phone, Email, IM

Know who is in the room

It is much easier to tell who is talking in a face-to-face meeting then in a conference call. When someone on the call makes a good point and you don't know who was talking this problem is painfully clear. Taking notes will help you remember who is on the call and identify them later. Write down everyone's name and some descriptive hints. These notes will help prevent you from needing to ask "who said that?" later in the call.

Make sure you can hear them

Good quality conference phones are indispensable. With a good phone people don't have to shout. When people have to speak loudly they are much more likely to become annoyed and frustrated. A good conference phone will let people feel like they can really talk to you. Convince your team to invest in a conference phone with two-way speaking capabilities. It will also help them communicate well with customers and other teams.

Sample Conference Call Notes

- Alice (Engineer) - high voice, sits close to the phone
- Bob (Doc writer) - speaks quickly and clearly
- Carol (Project Manager) - Southern accent, speaks loudly
- Dave (Engineer) - quiet speaker with a slow voice

Make sure they can hear you

Invest in a good quality phone and use it over a land line. Higher quality phones have much better fidelity than the cheaper ones. It is worth spending money on a land line and a good phone so your voice doesn't sound fuzzy and tinny. Headset phones will save your neck and let you type while you are on the phone[1].

A good phone is an important tool, but you need to use it well. Speaking to a room full of people over a speakerphone is very similar to talking to an audience from a stage. You need to speak in a full and clear voice. Don't yell, don't mumble, and make sure to enunciate properly. And make sure to pause when you speak so other people can follow you more easily.

I like to choose someone to emulate. Television and movies are a good place to find a model. Actors take lessons on proper presence and enunciation. Pick someone dignified and respectable and base your phone "performance" on theirs. I am a fan of *Frasier* star Kelsey Grammar, but you have to choose someone who fits you. If you have a naturally high-pitched voice then don't try to emulate James Earl Jones.

Remove unwanted audio distractions

Don't let distracting noises get in the way of your call. Machine noises, fans, air conditioners, and other noises from the conference room can make a conference call untenable. The simple solution is to move the speaker phone. If you are talking to a room full of people using a projector, then have them put the phone on the other side of the table. If someone is having a secondary conversation you should ask them to move away from the phone so you can hear the primary speaker.

Watch out for noises coming from your side of the conversation as well. Your kids playing, your dog barking, or even the sound of your footsteps as you walk around can be very distracting. Use the mute button whenever you aren't speaking to avoid interrupting the call with background noise.

Talking About Time

When you schedule a conference call, or other meeting, be specific when you talk about time. It isn't enough to say Thursday at 10. You need to say Thursday September 25 at 10:00 AM EDT (Eastern Daylight Time). Remember, 10:00 isn't the same time for everyone.

Pause

Conference calls lack the simple visual queues that someone wants to talk. They can't raise their hand or make eye contact with the speaker. This can lead to people getting shut out of the conversation. If you are running the conference call make sure to pause the conversation from time to time and give other people a chance to speak.

If you aren't running the call this can be tricky. Many time the call leader will be shutting people out without thinking about it. If you hear someone trying to speak and getting shut out then be their advocate. Say, "I think Fred had

1 I like Plantronics phones and headsets www.plantronics.com

something to say." Two voices are much easier to hear than one.

Little Distractions Replace Big Ones

During my first long conference calls I was very easily distracted. I never had this problem in the office since there was a lot of visual input to keep me focused. But over the phone long team meetings were deadly boring. This was a big problem since I really wanted to pay attention to the meeting.

I tried everything I could think of to remove all distractions. I closed the window shades, turned off my computer, and removed anything I could play with from my office. Nothing worked. The more I removed distractions the more distracted I would get. Around 90 minutes I would always start wandering around the room looking for something to occupy my excess attention. When that stopped working I expanded my wanderings to other areas of the house.

On one ill-fated day, in the middle of the third hour of a conference call, my eyes fell on an a box of supplies my wife had ordered for a craft project. With a glee I can only attribute to hours of staring at a blank wall, I opened the box and found it was filled with enough random junk to occupy me for the rest of the day.

I played idly with the spools of wire and small plastic tubes until I noticed a small deflated beach ball. This was perfect. I could blow it up and play with it for the rest of the meeting. Sure I didn't have a mute button, but I was certain I could inflate the beach ball quietly and nobody would notice.

The plan worked great until I got to the end. My tongue was pressed into the plastic valve and I needed to move my finger to replace it over the hole so I could stop the air from escaping long enough to get the plug in. The maneuver seemed simple enough, but someone asked me a question just at the wrong moment and in my haste to answer the valve slipped causing a loud wet plllfffftttt noise right by the mouth piece of the phone.

The meeting stopped dead and after a pregnant pause a teammate asked me, "Zack, did you just blow us a kiss?"" I stammered something about having sneezed and quickly moved to another topic. To this day I am cenrtain that nobody knew what was really happening and not a single person in the room believed the sneezing story.

Now I always keep a small Hacky Sack by the phone so I have something to do with my hands during long calls. I also bought a phone with an easy to use mute button. It was money well spent.

Be polite

You don't have to yell to make yourself heard. Conference calls can be frustrating. I occasionally get the urge to tell everyone else to shut up and listen to me. The problem is, when you get frustrated you will stop listening to other people. However well you might think you are hiding it, everybody knows what you are doing. Cutting people off and bullying your way into the conversation will not help you.

The goal is to make the other people really listen to you, not to be the loudest person in the room. Take a deep breath and realize you will get your chance to speak. Give your coworkers the benefit of the doubt. They are just trying to be heard as well. Don't interrupt people. Be assertive when you need to be, but do it politely.

Avoid Long Distance Charges

Save strain on your patience and your bank account by avoiding long distance charges. Many companies in the United States offer service plans with unlimited long distance for a flat fee. They are a good investment.

Always have a second option for talking to the people in the room

Sometimes phone connections get lost. Don't let it derail your conference call. Have a second line of communication to the other people on the call; like IM or IRC. You need a simple way to say that you need them to call you again or that you are speaking and they can't hear you. Ideally you can resolve these issues with one person's help instead of disrupting the whole call.

Video Conferencing

Dr. Paul Ekman is the science consultant for the television show *Lie To Me*. It stars Tim Roth, the guy robbing the diner in the movie *Pulp Fiction* and the bad guy in *The Incredible Hulk*. Dr. Ekman teaches Mr. Roth, and the rest of the cast, how to lie convincingly.

A lot of the way to lie successfully is in your face. The way your lip curls up at the edge or your eyes wrinkle show if you are lying or not. Dr. Ekman teaches these actors how to lie by looking at their faces and talking to them. He studies the minute twitches and spasms of their facial muscles and coaches them about how to be more convincing. And he can do it over Skype's video chat.

Dr. Ekman uses video conferencing, but almost none of the teams I spoke with use it. A study in the Harvard Business Review found that over two-thirds of the teams they sampled found video conferencing "too fuzzy to enhance the collaboration experience."[1]

1 Can Absence Make the Team Grow Stronger by Ann Majchrzak, Arvind Malhotra, Jeffrey Stamps, Jessica Lipnack

When Dr. Ekman coaches actors he makes the face of the person he is talking to the same size as it would be in real life. Dr. Ekman' years of experience have shown him that we relate better when someone's face looks like it does in real life.

When it works

Video conferencing is ideal when you need to convince one person of something difficult that takes a lot of trust. Seeing your face helps people trust you. Just watch out for the Mike Teevee problem.

Mike Teevee was one of the children in *Willy Wonka and the Chocolate Factory*. Mike sent himself through a television signal and came out in a TV on the other side... three inches tall. Willy Wonka was only able to pull Mike back to his normal size because "fortunately children are very stretchy."

The Mike Teevee problem is common. Take a man who is two meters tall and send his picture of him over iChat and he becomes a two-inch window. Video conferencing is very distracting, on purpose. When you really need to see someone's face make it large on your screen and ask them to do the same.

When it doesn't work

Neil Stephenson's ideas were a major influence on the virtual reality program Second Life.

In the book *Snow Crash*, Neil Stephenson describes a piece of technology called the MetaVerse. The MetaVerse is an advanced form of chat room where people are represented by ultra realistic human forms. This chat room becomes so useful that people don't travel much anymore. They conduct business meetings, have dates, and generally interact in this virtual reality.

The success of holding meetings in this virtual world comes down to the ability to see everyone in the room in depth and in focus. Being able to look from one person to another is the big difference between a truly useful system and the video conferencing we have today. There are advanced video conferencing systems like HP's Halo system, but the high cost of these systems and the need for a dedicated connection makes them unusable for most companies.

In the next 10 years basic video conferencing may advance to the point that it becomes useful and cost effective. Until then it isn't very useful for communicating with groups.

Video conferencing is about trust. When you need someone to see your face and trust you it may be helpful. Most of the time just pick up the phone or send an email.

How You Communicate Matters

The way you communicate with your team shapes their idea of you as a person. How you come across on a conference call or an instant message makes a big impression. It can make the difference between someone trusting you or not.

Writing code, creating products, and fixing bugs can't happen unless you communicate with your team. Make that communication as easy as possible.

Help your teammates stay connected to you by keeping it simple. Send someone the conference call number instead of making them look it up. Let them know if you won't answer their IM for a little while.

How you communicate is just as important as what you communicate. Remove any concern your team might have about working remotely with good communication.

Written Communication

Writing clearly is the most important skill for a remote engineer. When you work remotely the majority of your communication with your team is written. This writing takes place in many formats. The best way to write for a bug comment is not the best way to write a design document or an email.

This chapter focuses on two areas: how to write more clearly and where to write. If you are want to improve your performance as a remote engineer, then improving your writing is the first place to start.

No matter who you're communicating with, all good writing has consistent features, so I'll touch on how to make your writing better in general first, then I'll detail how each type of remote written communication differs from normal writing and how to write best in that medium. Most importantly, we'll talk about when you should be using each type of communication, and for what.

General Communication Guidelines

Every form of communication in this chapter is archived. Practically speaking, what you write in a bug report or on a wiki page will be available forever. All of the systems in this chapter are visible to other people, whether it is your company reading email logs or a future employer reading your blog.

Choose your words carefully and consider the following general guidelines.

- Stick with the facts. If there is a controversial issue, avoid giving unsupported opinions. Take the time to make a reasoned argument based on facts.
- Be respectful. It is much easier to offend people when you don't have a personal relationship.
- Be polite. You will never regret being too polite.
- Avoid profanity. Remember, what you write may be seen by your coworkers, customers, and potential employers. Keep everything you say safe for work.
- Check your spelling and grammar. A few typos are unavoidable, but you should take the time to use a spell checker and proofread your comments at least once.
- Stay off the soap box. If you are having a problem with a person or policy in your workplace you shouldn't bring it up in public. A bug comment or wiki page is not the right place to air your personal grievances.
- Stay positive. Bringing up issues is fine, but focus on solutions rather than problems. Make suggestions, not just criticisms.
- Be very careful about privileged information. Never post privileged information to public systems. If you have any questions about a specific piece of information you should ask your manager.

Improve Your Written Communication

Emails, wiki page, bug reports, and design documents are all referred to as documents in this chapter.

Karl Fogel, a founder of the Subversion project, puts good writing at the top of his list of skills required in a remote engineer. You can post a brilliant idea to a mailing list, but if it is poorly written "then nobody is going to read it and it won't do any good. It doesn't matter how brilliant a designer you are."

Clear and descriptive written communication is one of the defining skills of a remote engineer. People will judge you not only on your ideas, but by how well you convey them in an email or a bug report. Persistent forms of communication require more consideration than an instant message. Your email should reflect your written style rather than your spoken one. The first step is to review everything you write before you send or post it.

Think About Your Audience

Everything you write should have a specific audience in mind. A mailing list may include just one coworker or everyone in the company. When you write always think about what you are writing from your audience's point of view.

When you think about your audience's point of view remember that they don't know everything that you know. This is a classic tip for better writing and most engineers I know are really bad at it. Use citations a links to help your audience understand what you are talking about. Any time you mention a bug number, a specification, a previous conversation, a different article, or a different document link to it. A link takes up very little space in your document and will really help people follow what you are saying.

Read What You Write

You should read every wiki page, bug report, and blog entry before you post it. Look at the whole document and make sure it says what you meant to say. Consider the people you are writing for and make sure they will understand you.

Another reason for reading your documents again is to look for typos and grammar problems. You will never catch them all, but you want to keep them to a minimum. If you have the privacy, reading the document out loud helps you find errors quickly. If something sounds wrong it probably is.

Grammar Girl

Improving your writing will make a big difference in how well you can convey your ideas to the team. The Grammar Girl podcast is a good place to start. If you have ever wondered if you should use "effect" or "affect," or the difference between "who" and "whom," Grammar Girl can help. Check it out at http://grammar.quickanddirtytips.com.

Read What Others Write

Reading will make you a better writer. You need to read if you want to write well. Looking at the way someone else expresses their ideas will help you express yours. Books are good, so are magazines, technical documents, and longer blog articles. Longer documents with reasoned arguments will help you the most. Short blog articles and micro-blogs can help as well.

Reading almost anything will make you a better writer overall, but you should also read examples of the specific kinds of documents you are writing. If you are writing a wiki page then read other wiki pages for that project. For a design document, read the other design documents for that product. Your writing will be more successful if it uses the same style as other writing from that team.

Use Punctuation and Capitalization

Proper capitalization and punctuation will make your documents easier to read and understand. The goal of your writing should be communicating your ideas. If your reader must decipher bad punctuation and words without capitalization, they will be distracted from your content.

Style

Everyone has a writing style of their own. You may write simple sentences. Or you may prefer to write long and grandiose sentences, full of elaborate subclauses, with complex structures, complete with additional exposition. You should focus on being easy to read with simple sentences that make your points. You don't need to show off. Overly simplistic is always better than difficult to understand.

Don't Be Funny

Wit and especially sarcasm often depends on non-verbal cues. They can easily become blunt or rude when written down. Documents lack all of the subtle nuance of interpersonal communication. Just focus on saying what you want to say. Be conservative with your choice of jokes and language, and don't use profanity.

The Curious Case of the Misplaced Modifier

If you prefer to read a book rather than listen to a podcast you should check out *The Curious Case of the Misplaced Modifier: How to Solve the Mysteries of Weak Writing* by Bonnie Trenga. This book provides simple examples and exercises which can help anyone be a clearer and more compelling writer.

Direct Email

Kim Rose, the co-founder and Executive Director of Viewpoints Research Institute, uses email in a way that borders on real-time communication. She doesn't use IM or IRC, but her coworkers would notice if she went much more than an hour without sending an email. It keeps her as connected as a real-time tool.

The email you send to one or two people is the closest thing to real-time communication of any of the systems discussed in this chapter. Email is the only form of communication in this chapter where you are deliberately choosing your audience. This is in contrast with a blog or forum where you post information and the reader determines if they are interested. Addressing a small select audience requires a different style of writing than addressing a larger audience.

Email is also one of the easiest forms of written communication. When you send an email you don't need to find the right forum or format a wiki page: just type your message, review it, and send it. Email is the perfect communication mechanism for specific non-urgent questions.

Use Email By Default

Want more information about writing better email? Check out Send: Why People Email So Badly and How to Do It Better by David Shipley and Will Schwalbe

Emailing someone is the least intrusive way to ask them a question. It lets them continue their work uninterrupted until they are ready to respond. Email should be your default choice when you have a question for a teammate.

Answer Emails Promptly

Lost connections, SPAM filters, and accidental deletions can make an email disappear. When you get an email from someone you should answer them quickly so they don't worry that the email was lost. A good rule of thumb is to answer emails from your team within two hours during normal business hours.

A Word of Caution

> "Many in our study found e-mail a poor way for teams as a whole to collaborate. They reported what others have noticed as well: Trying to do the main work of the team through one-to-one exchanges between members can cause those not included to feel left out, diminishing the trust in the group and leading ultimately to dysfunction."[1]

Email is an exclusionary form of communication. When you send someone an email it is not visible to anyone else. Private communication like this works well for some topics, but not for collaboration with a team.

The personal nature of direct email makes it a good place to communication directly with a very small group of individuals. Direct email is a tool for communication, not collaboration. Using email to support mailing lists can be an excellent way to foster collaboration.

Mailing Lists

Mailing lists are a hybrid form of communication. They share many of the group communication dynamics of wiki pages or forum postings without having the same sense of a shared space. They have no mechanism for editing content

1 *Can Absence Make the Team Grow Stronger* by Ann Majchrzak, Arvind Malhotra, Jeffrey Stamps, Jessica Lipnack

after it has been distributed. However, dealing with groups in an email list means that you have to consider a few extra details.

Send To The Right List

Make sure you are sending your email to the right list. Most mailing lists have a specific subject. This might be a knowledge-based area like "developers of a specific tool," a geography-oriented subject like "everyone who sits on the third floor," or something else entirely. When you post to a mailing list you should stay on topic.

Mailing Lists For The Whole Company

To help organize their large volume of email the Viewpoints Reasearch Institute team uses mailing lists, but not in the standard way. Normally mailing lists are used to communicate to a specific subset of people. VPRI uses mailing lists to categorize information. Every member of VPRI is on every mailing list and the list name is automatically prepended to the subject line of every email. This makes it very easy for team members to ignore email they aren't interested in.

Using mailing lists to provide information to the entire group in a way that keeps everyone informed but can be easily ignored is a lot like IRC. Some members of the VPRI group are moving toward IRC, but it is a slow process.

Use Threads Properly

Mailing lists rely heavily on the concept of threads, which organize emails by topic. Threads start with a specific question and continue until the issue has been resolved or everyone gets bored. Which thread an email belongs to is determined by the subject of the email. When you reply to an email most mail programs will add "Re: " before the subject line and automatically include your email in the thread.

The goal of a thread is to create a coherent trail of information about how a decision was made and make it easy to find that trail in the mailing list archive. Keeping the thread useful means staying within the designated topic of that thread. If you want to discuss something else you should create a new thread by sending an email with a new subject.

Mailing list archives

Many mailing lists are archived and made available on a website. These archives can be years long and include tens of thousands of messages. A well-organized mailing list is an asset to your team, but a disorganized one just takes up disk space. When you post to a mailing list you are adding content to this archive and you should make sure your contribution will enhance its utility. You will benefit from thinking about mailing lists this way even when they aren't archived.

Forums and Newsgroups

Forums and newsgroups are an industry-standard way for engineers to communicate with customers. They work very similarly to archived mailing lists. Forums are an archiving non-editable form of communication. What you write will be there forever and you probably can't edit it. They are also very public. When you communicate with a customer on a forum you are the voice of your company. A single good interaction with an engineer can create a loyal customer and a bad one can send them to your competition.

Clear and Focused Subject Lines

Most forums are display lists of subjects separate into specific categories. A reader will choose whether to read your posting based almost entirely on the subject. Make your subjects descriptive, short, and very focused.

Small, Transient, and Readable

Forums have basically no formatting and very limited context which means your writing should be small, transient, and readable. Forums are not the place to post design documents or explore large and complex ideas. A forum may feel like a wiki page if you post to it with your web browser, but it is a lot more like a mailing list where postings are quickly obsolete. Your forum post should be short enough that it only takes a few moments to read.

Forums and newsgroups are a transient form of communication. Your post on a forum will last a few days at most before new content pushes it out. Focus your forum writing on short-term topics. "What should we do for this dialog?" is much better than "What should our new product look like?"

User Forums Make You More Visible

User forums will make your customers feel more connected to your product and your company. They are a good place for them to get support for your products. Customers will appreciate the time you spend helping them in the forums. It will also help your career.

As a remote engineer, staying visible within your company is a constant struggle. The people outside your team may not even know your name. When you help customers you are building customer loyalty to you personally as well as to your company. Your company's management will notice this customer loyalty.

Blogs

Blogs are flexible enough to defy categorization. Most blogs function as single author, one-time edit, open ended public systems. Readers can't edit a blog, but they can add content in the form of comments. Blogs are a good place to give the people outside of your team a view of what you are doing. Design documents, systems documentation, and other reference materials will get lost here because blogs are organized chronologically rather than by subject. Store reference documents on a system that gives you more control over the organizational structure, like a wiki.

Blogs As Team Communication

A planet is a specialized type of blog that focuses on a specific project community and has a small group of editors. Planet blogs normally have a small group of editors. Mozilla uses planet blogs for many of their project communities.

The Mozilla Labs Ubiquity team relies heavily on blogs for team communication. Many new UI concepts and features start as blog posts. The team debates new features in the blog post's comments. Each team member currently maintains their own blog. They are working on creating a planet blog for the entire Ubiquity team so there will be one place for all of the team discussions.

When a team member suggests a new feature it normally starts with a picture of the feature and a couple of paragraphs describing why it is important. These descriptions often focus on a general type of user interaction rather than a specific product feature. For example, a recent post showed a working example of a new style of popup menu with discussions of why it is more intuitive and easier to use.

Blogs help the team communicate about features, but they also help the team educate potential team members about user experience design. Teaching about the process of design is a major focus of the Ubiquity team.

Aza Raskin the head of user experience at Mozilla Labs communicates many of his design ideas by posting videos to his blog that show specific features or ideas. Aza's videos are impressive software demonstrations. He frames the discussion, shows how the features works, and talks about why it is important. He typically does all of this in under five minutes.

So... should you start a blog? This is a tricky issue but in the end it comes down to one simple question: do you have something you want to say that you can't say as well in any of the other formats, and do you want to say it publicly? A blog can be a good format for reporting your status. You can keep a running narrative of what you are working on and the problems you are solving. This is a good idea for keeping your team in the loop and a useful tool during performance reviews.

Micro-Blogs

Micro-blogs are a specialized type of blog which encourage very short updates (normally less than 140 characters). Twitter is the most well known micro-blog, but many programs support features like micro-blogging. Choosing to micro-blog is a question of preference. Many professionals keep short notes on the tasks they are performing throughout the day. These include simple announcements of technologies they are trying, problems they are solving, and places where they are getting stuck.

The key to using micro-blogs well is understanding what they are and what they aren't. When one of your coworkers wants to find out what you did that day a micro-blog can be very helpful. Simple messages like *finally fixed bug number 1138* or *just started looking into a new unit testing solution* do a good job of telling people what you are focused on. The good part about micro-blogs is that they are archived and non-disruptive. People can catch up at the end of the day without interrupting their work.

Micro-blogs are not a good place to share "must read" information. The short messages in micro-blogs can become overwhelming and people could easily miss something. If you post regulary then each message will probably be onto the next page in less than a day.

Making Your Blog Successful

A professional blog is a good way to let other people see what you are doing and what you are thinking, and showing people how you work is very important for remote engineers. A professional blog is also a lot of work. Should you start a professional blog? If you have the interest, the time, and enough to talk about, then it probably can't hurt. If you do start blogging you should adhere to the following guidelines.

Keep it clear and concise. More content is not always better. Shorten your blog posts and remove any content that isn't helping the post.

Lead with your main point. Every blog post should have a clear thesis in the first paragraph.

Stick with your areas of expertise. Don't spend time ruminating on random topics. This is your professional blog; you should write about your profession. Stick to areas where you can show off your real expertise.

Focus on the facts. Your blog is not the place to mull over the big questions of the universe. These posts will rarely come out well. It is a lot easier to write well about specific facts.

Keep it professional. Your blog will live on the Internet forever. Other people will link to it and search engines will cache it. When you post something to your blog remember that a future employer might read it. All of the general guidelines for written communication apply.

Don't reveal privileged information. This is important for every public system in this chapter, but it is especially easy to accidentally reveal information in your blog. When you write about what you are working on, you must be careful not to disclose information about upcoming products or internal policies that are not public knowledge.

Personal Blogs

A personal blog can be a fun way to express yourself and share your opinions with other people, but don't mistake them for a professional blog. Keep your professional blog posts completely separate from your personal ones. Make sure the two do not show up on the same screen.

Even though you separate your personal and professional blogs your coworkers, customers, and potential employers could still find your personal blog. Before you write something in your personal blog ask yourself how you would feel if your boss read it.

These tips were strongly influenced by Chris Brogan

Bug Tracking As Communication

The Mozilla Labs Ubiquity team tracks bugs with the Trac issue tracker, and they also use it to educate their team about user experience design. As an open source project, Ubiquity gets contributions from people outside of the team in the form of enhancement requests and patches. Each patch needs to be evaluated to make sure that it fits with the existing user experience.

Enhancement requests cause debates. This debate gives the person entering the request a chance to understand the priorities of the team. They need to discuss the feature and how it will fit into the product before it will be added to the release. This process allows the Ubiquity team to educate potential new members about their particular views on user experience.

Your bug tracking system is probably the most versatile piece of software in your organization. It tracks bugs, but it also handles backlog management, task assignments, release planning, and design discussions. Systems like Bugzilla also support the coordination of code reviews and other team-oriented tasks.

Your bug tracking system is where you should talk about specific issues and specific solutions. Working remotely means you can't talk about a bug in the hallway so you will need to rely on your bug tracking system more than usual.

The comments in your bug tracking system can be used to communicate issues and review decisions. The bug tracking system is also another opportunity to show other people how you work. Writing more of a comment than "fixed" will give you the chance to talk about what you changed and why you changed it.

When you write comments in a bug it is important to talk about the changes you made, but also why you made them. Explain your thinking about the way you fixed a bug. Why did you change this particular API or UI the way you did? You should also reference the files you changed when making the bug fix.

- Add a descriptive comment with every change.
- Talk about why you are making the change.
- Reference the code you are changing.
- Write directly to the person who wrote the bug.

Version Control

Version control is an often-overlooked avenue for the exchange of ideas. The metadata stored in a version control system -- comments, diffs, and change logs -- presents an accurate representation of your coding effort. The bug tracking system is where you talk about your product and the version control system is where you talk about your code.

The purpose of a version control comment is the place to talk about the changes you made to the code and why you made them. Taking the time to put valuable information into version control comments will make the version control system an excellent reporting tool for your team, as well as increasing your visibility.

Version Control As a Reporting Tool

Version control is a single place where you can see all the changes the team has made to the code over a fixed period of time. All version control systems have reporting tools which can show you the logs of what has changed. These logs differ a little bit between one product and another, but they all have the following information about each change: when it was made, who made it, what files they changed, and the comment they made. The information in these logs flows in two directions. The log will help you see what your team is doing, and it will help your team (and your manager) see what you are doing.

Encourage your team to write descriptive check-in comments by setting a good example and showing them how useful this information can be. There is no better tool for getting a good understanding of the code changes that are being made.

Check in often

Checking in your code is how you share it with the rest of your team. Part of sharing information proactively is checking in often. A good general guide is to check in as soon as your code won't block other people. This should be multiple times every day. Checking in often not only shares your code with your team, but it gives you a good history of your work.

In order to accommodate this style of checking in you should strongly encourage your team to work with branches. Version control branches are a way of separating the code you are working on from the code everyone else is working on. A standard use of branches is to quarantine a product just about to be released. Most products go into a code freeze at the end of the release cycle and branches allow you to keep working during this time.

Learn The Command Line

Most version control systems have a combination of graphical and command line tools. Learning the command line can save you a lot of time when working over a remote connection. Almost all version control tools will run faster in the command line since you can be more specific about what you want the tool to do. A little time spent learning the command line will save you a lot of time in the long run.

Wiki

Wiki is a quick and easy system for creating, linking, and categorizing reference documents. It is available to everyone, editable by everyone, and tracks the history of changes made to each page. Wiki is a freeform structure; you and your team can fill it up any way you want. This flexibility is the greatest strength and the greatest weakness of wiki.

Wiki systems are well suited to documents and document-style information. They are a good place to share design documents, procedural documents, and meeting notes. When a new member joins your team you should be able to point them to your team wiki as the first place to look for information.

Wikis need naming conventions

With a forum or a mailing list the subject line of each post can be more relaxed because they are stored chronologically. Reusing an email subject from a year ago is fine because most people will have forgotten the older email. Naming conventions are key to keeping your wiki manageable because each wiki page is persistent and requires a unique name. Wiki page names must follow a naming convention or they will quickly become impossible to manage.

When your team starts a new wiki you should strongly encourage them to create a naming convention for the wiki pages. A good page naming convention for a wiki should specify the following topics at a minimum:

- Upper and lower case letters - MyDesign vs. Mydesign vs. mydesign
- Group names as prefixes - MyGroupMyPage vs. MyGroup_MyPage vs. MyPage
- Agreed names for common terms - e-mail vs. email
- Words to avoid - For example, it is generally a good idea to avoid your company name since all the pages are about your company
- A common language for page names - It is much easier if you choose a language that can be represented using English only characters

Wikis need constant maintenance

When you add a new page you need to think about how it fits in with the rest of the information in the wiki. Make careful choices about where the page should go, what it should be named, and where it should be linked from. Make some plan and stick to it. A bad plan is better than no plan at all.

Sticking with your plan will be a constant effort with your wiki. Some wiki packages provide tools to automatically detect certain issues like orphaned pages, but you and your team will need to continually edit and refine the contents of your wiki or it will become disorganized and useless. A well-planned wiki will be a benefit to your team; a poorly-planned wiki will take a lot of time and effort without giving you much in return.

Your Personal Wiki Page

Most wikis have a page for each user. You should take advantage of this page to post some information about yourself. It doesn't need to be long, just a couple of paragraphs about who you are and what you do. Include a picture of yourself and links to any professional websites or blogs you have.

The Consolidation of Information

This chapter covered seven different methods for collaborative communication. There are many others available and there will be even more in the future. Every system has specific strengths and weaknesses. You need to choose the best system for each piece of information you want to share with your team. The goal is to establish a unified body of information across all of these systems. This will create a well-managed easy-to-use information source supporting your strong voice in the team. Manage your information improperly and you will just have useless noise.

Every team member is responsible for making these systems work, but these systems are more important to you because you are remote. The written communication mechanisms your team uses will help you stay connected. They are vital to your success as a remote engineer.

The only way to make these systems work together is to consolidate information. Adding new information is easy; even a small team can create an overwhelming amount of information in a very short time. If the knowledge in these systems isn't managed, then it will become redundant, overly complex, and useless.

Michael Pilato from the Subversion project described this phenomenon when he was discussing wikis:

> "The very power that [wikis] give you in being able to quickly create a category of information is the very thing that [causes the problem]... Eventually you can't find anything because the granularity is ridiculous."

Create metadata. Every time you create a wiki page, blog posting, or other shared document it needs metadata that describes what is in the document. This can be a short email subject, a descriptive introductory paragraph, or a formal document abstract. Forcing yourself to stick within the boundaries of your stated topic will also make a big improvement to your writing.

Organize in larger groups before smaller ones. When organizing the data of your team you should start with a few large categories. For each system you have to decide what it should be used for. You can drive this discussion within your team. Once you have to decided on the larger categories you can further categorize within each of them.

Create naming conventions. Your team has to decide on naming conventions to govern mailing list subjects, wiki page names, and bug descriptions. You have to push them to standardize. It doesn't matter what naming convention you choose as long as you stick with it.

Sign your name. Make it really clear what you write. Put your name on your documents in a prominent location. If someone else helped you, put both of your names.

Don't create redundant information. Before you add any new information you should make sure it isn't there already. Linking to existing information will enhance that information, but copying it will create redundancies which make the system more difficult to manage.

Continually refine. Following all of these guidelines will help a lot, but your systems will still become disorganized over time. You need to constantly reexamine your categories and documents and make changes to further improve the system.

Work with the tools, not against them

Every information storage solution has specific benefits and issues. Don't put design documents in your forums or try to communicate with your customers in your wiki. You can't use one system to solve all of your communication needs.

A wiki stores information differently than a bug tracking system or a mailing list archive. For every piece of information you share you need to choose the correct venue. Pick a method of communication that matches your content. Don't post a bug report on your wiki just because it is likely to get read.

The consolidation of information is a vital practice for any team. It will make the team more efficient and will make it much easier to introduce new team members. It is also an important step in removing impediments to communication. Maintaining a low cost of communication is critical to remote success.

Being remote means that you must exploit every means of communication available to you. Don't just talk on the phone. Taking part in the written communication of your team will connect you to team decisions and make you much more visible to your team, your company, and your customers.

Remote Presentations

Every presentation is an emotional journey.
-Nancy Duarte

Have you ever had a really good idea that your boss didn't listen to? Can you remember a innovative feature you worked hard on that nobody noticed? Your innovative ideas and features won't be recognized if you don't present them well.

You give presentations and demonstrations less often than someone from the marketing department, but yours have higher stakes. Many of the presentations you give will be in front of executives or customers. Those presentations will encompass larger issues: your achievements for the quarter, why your team should make an important decision. Demonstrations are always selling your product, either to customers for money or to other people in your company for recognition. The big demonstrations you give might be the only chance you get to interact with the executives in your company.

Interacting with your audience is more difficult when you present remotely. In person, if your audience is apprehensive you can take extra time to reassure them. If they are excited, move quickly to encourage their excitement. Reading your audience like this is just as important for remote presentations, but a lot more difficult.

You might prefer to give all presentations in person, but that isn't possible on a distributed team. Some remote engineers leave presenting to colleagues in the office. If you do that you are missing out. A presentation is the opportunity to show your work and ideas. Don't let it pass you by.

Creating Good Remote Presentations

Edward Tufte is well known for his excellent books and seminars on presenting data. I was lucky enough to attend one of his workshops where I was blown away by his presentation. He had compelling and educational material, but his style was even better. He was engaging and witty for his entire six-hour lecture. Tufte presented real data, engaged his audience, and varied his material in order to keep everyone interested.

On a different occasion a friend of mine gave a presentation about web technologies for Java. He began his presentation by announcing that he had two hours of material and only 90 minutes to present it. He then spoke without stopping for breath for the next hour and a half. He ended by announcing that he was out of time so he wouldn't be taking any questions. This kind of information overload is barely tolerable in person and will ensure that your entire audience has stopped paying attention by the end of your presentation.

Everyone has a unique style of presenting. Your goal is to take your style and adapt it so you leave your audience feeling relaxed, engaged, and informed.

What is desktop sharing software?

Desktop sharing software, otherwise known as web conferencing, allows you to share your desktop with a remote computer. That means they will see your desktop in a window on their own desktop. Web conferencing packages also include text chat, virtual whiteboards, and other collaboration features. Desktop sharing programs can be used to run presentations or view other peoples' presentations. Most desktop sharing programs can be run just by clicking a link in your web browser.

You can share with a computer in the conference room that projects your slides. Desktop sharing can be used to share Microsoft PowerPoint, Open Office, Apple Keynote, and any other software you are running. Web conferencing is also an effective collaboration tool for graphical tasks like designing user interfaces or presentation slides.

Giving presentations and software demonstrations is important to the career of any engineer, remote or otherwise. On an agile team you will give a demonstration of your new features at the end of every sprint. Presentations are also a powerful way to show your team what you are passionate about and get them engaged in what you are working on.

A presentation is a conversation and you need to keep your audience engaged. Talk with them, not at them. A good presentation is a lot more like a conversation than a lecture. Start every presentation by encouraging your audience to stop you and ask questions. Remind them of this at key points during your presentation. Many desktop sharing programs have a chat feature your audience can use to easily ask you questions.

Encourage people to speak up when they disagree with you. When an audience member has a comment about something you say, it is a good thing. It means they are engaged enough in your topic to really be thinking about what you are presenting. Always leave plenty of time for audience interaction and stay relaxed enough to handle interruptions during your presentation.

Presenting Well

Giving presentations is a skill that improves with practice. Over time your presentation skills will improve as long you remember the fundamental rule of good presentations: always think about your audience's point of view. You already understand the material. You are giving the presentation to benefit them, so present it clearly and put it in context.

Breathe. Presenting makes people nervous and nervous people rush. Being nervous is fine, but you need to pace yourself. Take a moment to breathe and slow down while you are speaking.

Enunciate. It doesn't matter how good your material is if nobody can understand you. Don't mumble and don't slur. Choose someone with a reassuring voice to emulate. Think about good presenters you have seen. Focus on how clearly they spoke every word. Don't make your audience work hard just to understand you.

Be confident. Everyone is there to listen to you. A good presentation needs to be authoritative and a big part of that is being confident. You should speak confidently, or at least fake it well.

Rehearse your material. Rehearsing will help you be more confident. When a stage actor recites a monologue they make it look as natural as a friendly chat, but it takes them many hours of practice. Rehearse in three ways: in front of a mirror, into a voice recorder, with a spouse. You don't always need this much preparation, but it is worth it for big presentations.

Be interactive. Encourage people to interrupt you and to ask questions. The more people ask you questions the more you know they are really listening to you.

Burst your content. Nancy Duarte, the author of *Slide:ology* says, "Most remote situations have a major distraction for attendees called their [email] InBox. It's pretty assured that attendees will not be solely focused on the presenters. We advise our clients to create intentional distractions in an effort to regain attention." Most people's attention starts to lapse after about 10 minutes (Brain Rules p. 74). That is a good time to burst your content by doing something a little different. Surprise people. Tell a story, show a video, do an activity: whatever it takes to make sure your audience stays engaged.

Ask questions before you answer them. Questions are a powerful tool to keep your audience engaged. It makes them think about the answer instead of waiting for you to give it to them. Start your presentation with a larger question and lead to the answer at the end. This can be much more interesting that just starting with the answer. It isn't always appropriate, but it works well in the right cases.

Stay on topic. Keep your subject in mind for your entire presentation. Don't get bogged down in details and don't let your audience pull you off to other topics.

Don't let someone else control your slides. You should be the one advancing the slides on the main copy of your presentation. Letting someone else control your slides just wastes time and energy while you coordinate with them. Don't interrupt your presentation every couple of minutes to say "next slide."

Have fewer presentations. Most groups have too many presentations. Don't spend the time and effort to prepare a presentation when a design document and an informal discussion will do the job.

Don't use a slide when you want a reference. When you are sharing reference material, like a list of links or a set of guidelines for your team, don't use slides. Slides are the wrong choice for information like this. Present reference information with a Wiki page, a text document, or another more permanent medium.

Presenting Remotely At PowerPoint Live

Garr Reynolds, author of the book Presentation Zen, gave the only remote presentation at the 2008 PowerPoint Live User Conference. He was part of the keynote and discussed giving professional presentations. Garr couldn't take the time out of his teaching schedule to travel to San Diego so he suggested a virtual presentation. The conference organizers agreed, but they were apprehensive: "I was pretty nervous about it. [The organizers] were very nervous about it."

Garr was the only virtual presenter that year and the organizers did a special setup for him. They set up an auditorium with two large screens and seating for 100 people. The screen on the left showed Garr's face with audio from Skype. The screen on the right was for his slides. Garr sent his slides to the person running the projector ahead of time to avoid technical difficulties. He wasn't driving the presentation.

The organizers also set up a camera in the room. They turned it so Garr could see some of the audience and a portion of the screen with his slides. It let him connect with the room and make sure he was on the correct slide.

To make his presentation work Garr shifted away from his normal fast-paced style with many slide changes. Garr's typical slides often contain just one word and having to say "next slide" every 15 seconds would have been unmanageable.

His first step for this remote presentation was to cut out or combine many of his slides. He made most of them a little more engaging than usual, but he still kept a fast pace. "One slide wasn't up there for more than 30 seconds or a minute at the most."

Garr also changed his speaking style. "It wasn't my normal performance." He made his presentation "very, very conversational. More of a show-and-tell kind of thing." He engaged even more with his audience than usual and talked more about specific images and slides. "More like Uncle Louie comes over and shows you photos of the Grand Canyon... In a fun way."

Garr was worried about giving a remote presentation, but he reports, "it actually was OK. It is sort of Zen in a way. It's not fancy, it's not slick, but it worked." Garr's presentation got rave reviews. He attracted more people than there were seats and his audience rated his presentation as one of the best in the conference.

"Why was it successful? It was not so much the content, it was because of the emotions of it. [The audience] enjoyed it. It was fun, and it kept their attention for 45 minutes."

Garr Reynolds made presenting remotely work for him. He changed his style and used the medium to his advantage. He had good slides and a good delivery. Most of all he connected with his audience, even though he was on the other side of the world. The experience left presentation expect Garr Reynolds convinced that "presenting remotely works."

"I didn't feel like I missed anything. In fact, I probably had more of an impact because it was different. It was the only remote presentation."

Garr's presentation at PowerPoint live succeeded because:

- He could see the audience.
- The audience could see him.
- His face was large enough to see emotions.
- His slides had more information and stayed up longer.
- He changed his speaking style to be more engaging.

Making Effective Slides

Your audience for a remote presentation is missing out on an enormous amount of information. They can't see you, your body language, or your facial cues nearly as well as if you were in the same room. You need to give them something else to concentrate on during the presentation or you will lose them. Show them information that will engage them. Part of engaging people is being concise and specific. Too many people put all of the information they are presenting in their slides. Your slides should complement what you are saying, not repeat it.

During remote presentations your slides are more of a presence in the room than you are. Let the slides speak for themselves. Your slides should provide useful data that is not part of your verbal presentation. Make your slides a little more complicated; it is OK that they take a little extra time to read.

Using Fewer Slides With More Data

The standard PowerPoint pattern is many slides with a little information per slide. This pattern is completely wrong for remote presentations. When you show information in such small chunks you are giving the audience just one way to look at it. You are also making them consume it at your speed.

Different people digest data in different ways and at different rates. Help your audience by making your data available in as many ways as possible. Give them more data in each slide so they can decide how they want to consume it.

Inspiring and Informative Presentations At TED

The TED conference (Technology Entertainment Design) is a place to get inspired and educated. TED is an annual conference focusing on "ideas worth spreading." Every presentation is recorded in high quality and made available for free on their website http://www.ted.com. Check it out to see some of the best presentations anywhere.

The first advantage of having more data per slide is that you don't need to change slides as often. Every time you change slides your audience will take a moment adjusting to the layout, presentation, and information on the new slide. This is time they are not listening to you or thinking about your data.

The second advantage is that you are giving each audience member the option of consuming the data in their own way. Maybe you see that your points follow a logical order from one to two and then to three, but this won't make sense to everyone. Some people want to start with the answer and then hear the question. Others prefer to start with the question and then look at the answer. If you add more data to each slide you can satisfy both types of people.

If you are presenting a successful product you might have quotes from customers showing how much they love the product. These quotes should be in your slides. Assume you have three quotes about the product. It might look something like this:

- "We saved $10 million in productivity." -Alice Anders, MegaCorp
- "Really easy to use. A great tool for productivity!" -Bob Baker, Colossus Partners
- "Our engineers learned this tool amazingly fast. It is great!" -Carol Conners, Titan Limited

Positive quotes like these can go into your presentation, but don't read them aloud. Show the slide and say, "customers love it!" Some audience members will read every quote, some of them will just scan the company names. Let them choose how much of this information they want and spend your time focusing on more exciting topics.

Mixing Up Your Content

Meaningful, insightful images or illustrations retain the attention [of your audience] longer. - Nancy Duarte

Varying the amount of data per slide will help your audience pay attention to your content. Change your pacing and rhythm. Some slides need more data for each slide, but not all of them. If you spend five minutes on one slide then show four slides in the next minute. After you show them quantitative data then help them remember the message with a simple overview.

Good quality images are required when creating a simple message. Nancy says, "Having something that is quick paced and has images that are meaningful makes your presence bigger in the room." Images with emotional content will help your audience remember your material. For a further discussion of this topic, look in Nancy's book in the further reading section.

People absorb information better if you give it to them with some structure. Structure your data to make is easier to read. Using more data per slide is not an excuse to abandon information sequencing.

Make each slide build on the previous one. Your slides should provide data that backs up and enhances what you are saying. Your talk should build one idea on top of another. This includes what you tell your audience as well as what you show them.

Don't read your slides. Your audience can read a sentence much faster than you can read it aloud for them. Native English speakers can read about 40 words in ten seconds (The Cognitive Style of PowerPoint). This is much faster than you want to speak during your presentation. Never ever just read your slides. It is a waste of people's time and it is deadly boring. You either have material to present that complements your slides, or you should skip the meeting and just send your slides.

In this example you are presenting on performance improvements you have recently made to your company's website. This type of quantitative data is difficult to present well. Assume you have some numbers that look like this:

- Initial page load time was 5 seconds: now less than one second.
- Website search time was 10 seconds: now 2 seconds.
- Used to support 100 hits per minute: now supports 1,000.

This is compelling data, but if you stand in front of people and read it they won't remember it 10 minutes later. Format this data for maximum readability and put it on your slides, but don't read it. Tell your audience a story instead. "I used to keep another window open so I could read while I waited for our site to load. Now it loads faster than I can click on it." Stories are much more memorable than raw data.

Giving Your Audience Something To Take With Them

Projectors typically have an even lower resolution than monitors. Higher resolution supports more data in the same space, but doesn't show up properly on presentation projectors. The solution is to give your audience control over the information. Whether they have digital copies or paper ones, everyone should have a copy of your presentation while you are speaking.

Nancy Duarte says, "People will multitask during your [remote] presentation. It's the nemesis of the medium. Instead of ignoring it, use it as motivation to communicate differently." When someone is reading your handouts they aren't listening to you. That's OK. Nobody in your audience will listen to every word you say. Don't try to force them. Let them go at their own pace. Some of them will want to skip ahead while others will need to go back to see what you were just talking about.

When presenting remotely it can be difficult to provide handouts. If your audience has laptops you can send your slides ahead of time and let each audience member follow along with them at their own pace. Many people will advise you that sending your slides out ahead of time is wrong, but it is the right thing to do when presenting remotely.

People are coming to see you, not just your slides. If your entire presentation is in your slides then you should just send them and skip the meeting or, even better, send a document instead of slides. However, your slides are a much larger part of your presentation when you are giving it remotely. There are three reasons to send your slides ahead of time.

Don't start your presentation by wasting time. Many meetings lose 10-20 minutes at the start with setup issues. Making your audience watch while someone sets up a projector or waits for your slides to come over email is bad start. Having the slides ahead of time will avoid many technical issues and meeting delays.

Let your audience go at their own pace. Your remote presentation slides should contain more information than the slides you would use for a face-to-face presentation. There will be a much larger difference in how long it takes the fastest and the slowest readers to finish each slide. Let everyone go at their own pace by giving them a copy of your slides.

Give your audience a preview. Many people are reluctant to attend a remote presentation. Your slides, and a good description of your topic, will give your audience a preview of what you are going to say and make it much more likely that they attend.

For high stakes presentations, give your audience something more to bring with them. Create a teaser for your presentation. Let them know why they should attend. Get people excited!

What's a demo script?

A demo script is a plan for your software demonstration. It can be a detailed description of everything you will show, or just a general outline. A good demo script will help you be more confident and relaxed during your demonstration while making sure you don't miss any points. When you don't have to hunt around for buttons or start over after selecting the wrong menu option it will make the software look more professional and your demonstration more convincing.

Avoiding Unnecessary Visual Distraction In Your Slides

Understanding a slide with a lot of visual junk is like trying to talk to someone next to a moving train. They may have something interesting to say, but you will never hear it over all the other noise. Everything you add to your slides either helps or hurts; nothing is neutral. Go over your slides and remove everything that isn't helping to present the information. Use images to provide useful content and not decoration. Your audience cares about what you have to say, not how pretty your slides are.

Examine each picture or symbol in the slide and ask if it really needs to be there. Remove unnecessary lines and drawings. Most bullets can be removed without changing the data. There is no need to include your company logo on every slide. Having some empty space on each slide creates visual balance.

Get away from corporate templates. The goal of corporate slide templates it to make every presentation look the same. They often include corporate branding on every slide. This doesn't help your presentation or your company.

Educating, Impressing, and Inspiring Demos

A software demo is a special type of presentation. Instead of slides, this presentation is organized around a piece of software. The first goal of a good software demo is to educate your audience. You have something they probably haven't seen before and you need to let them know what it is and how it works. While you are educating them you also want to impress them with good-looking software. The final step is to inspire them. When you show executives a prototype the goal is to inspire them. Make them believe in the potential of your project. When you show the finished product to your customers you want to nurture the understanding that this software will solve their problems.

Inspiring your audience means more than just showing them functionality. If you are showing a dialog box, don't just click through the dialog; talk about what it means and why they should care. You have to put everything into context. Talk about why you made the changes or added the features that you did. How will these features be used? Why will your customers care about them?

Demo Videos

Aza Raskin, from Mozilla Labs, uses software demonstrations to show features to the entire world. He creates short videos, normally just a few minutes, and posts them publicly. These videos are excellent examples software demonstration.

Aza focuses on specific features, discusses how those features will affect users, and inspires other people to be interested in them. You can watch his demonstration videos for free on his blog http://www.azarask.in.

Create a demo script. You don't have to map out every minute, but you should have at least a general understanding of what you want to say.

Stabilize your software. Don't fix bugs or upgrade your system an hour before your demo. Create a stable working environment for your presentation, and then leave it alone.

Know your audience. Don't show code to the head of marketing or APIs to a business manager. Make sure you show the right material to the right people.

Drive your own demo. As for any remote presentation, you need to be in control of your software when giving a remote software demonstration. Use a desktop sharing program so you can control the software while you are speaking.

Practice all presentations ahead of time. In many ways a presentation is like a play; all plays have rehearsals. Go through your entire presentation at least once before you give it. Look at every slide. Try every feature you are going to show. When you present in front of your audience make every movement with practiced ease.

Practice using the presentation software as well. If you are using a desktop sharing program then rehearse with it. Set up a small conference and present to a second computer. Using one computer to drive your presentation and one to watch it will let you know how your audience feels. It is a chance to sit in the audience of your presentation. This small conference is your opportunity to learn the software. Never use any tool for the first time in front of an audience.

Sharing The Screen

Should the audience see you or your slides? Imagine a typical presentation. The presenter stands in front of a large screen with their slides. They can walk around and speak to different parts of the audience, have different expressions, and make important gestures. You can't fit all of that visual information on a laptop screen. Ideally you would always use two screens like Garr did, but that isn't always possible. Most of the time you can show your audience you or your slides, not both.

The foundation of Nancy Duarte's celebrated presentations is her belief that "every presentation is an emotional journey." Decide what journey you want to take your audience on and it will help you decide if they need to see you, your slides, or both.

Showing Your Face

Nancy says, "If it's a difficult conversation where you feel that the number one thing they need to feel from you is trust, it's probably better that they have eye contact with you." When your presentation is emotional then your audience needs to see you; show them your face. Get a good quality camera and get really close to it.

Show them your face, your head and your shoulders. Let them see your hands too. Look directly into the camera and make eye contact. Showing your face will help you build trust with your audience.

When you do show your face, switch between that and your slides. Don't try to show yourself and your slides at the same time. There isn't enough room.

Showing Your Slides

When you are presenting data then show your data. Make your slides fill the whole screen. Showing your face takes space away from your slides. Your audience needs to hear what you have to say; they don't have to see you say it.

Think about your emotional connection with your audience. If they know you and not your argument then show them your data. If they don't know you let them see you. When you are making a difficult argument convince them with your presence as well as your voice.

- Use fewer slides with more data
- Burst your content
- Encourage questions

- Let everyone go at their own pace
- Control your own slides

Making Your Virtual Presentations Real

Face time with your team, even virtual face time, is important. Being remote means you need to step up and take credit for your work. Giving presentations and software demos is a way to help your team, your boss, and your customers recognize your work.

A good presentation will be a good remote presentation. A mediocre presentation will become bad and a bad one will be abysmal. When you aren't in the room your material stands on its own. People won't remember what a good speaker you are. They will hopefully your arguments and what you had to tell them.

Always remember your audience. They aren't in front of you, but they are listening and watching you. Nancy Duarte uses visual aids to help her *see* her audience. She describes, "I draped off the windows of my office (because I felt silly), posted pictures of employees (so it felt like a real audience), stood up, clicker-in-hand, and delivered the presentation."

Do whatever you need to make the audience real for you. Stand up so you can speak with a loud clear voice. Walk around so they can all *see* you. Use physical movement and visual cues to help yourself engage emotionally. And smile. Nobody will see you smile, but they will all hear it. When you smile your voice becomes lighter and more engaging.

Giving a good presentation is almost always more work than you think it will be. You need to decide what you are going to say, prepare your presentation materials, and practice your performance. The work you do planning ahead will make a big difference to the quality of the end result.

Giving good presentations is important to your career and you can give a good presentation remotely. It just takes a little more planning and an understanding of some presentation basics. Take every opportunity you can to give presentations. Being in front of people is the difference between getting noticed and being forgotten.

Team Profile: 37signals

Our products do less than the competition —intentionally.
- the 37signals motto

37signals is a small software company based in Chicago; led by their founder, Jason Fried, and managing partner David Heinemeier Hansson. Jason and David are half-time managers at most. They spend the rest of their time creating products. Their management philosophy is that the best working environments comes when the managers are also workers.

The company has an office in Chicago, but the entire company works remotely. Only about half the staff is in the Chicago area, and they only come into the office one day a week. Everyone else is located in home offices around the United States and Europe. When David needed to work from Sweden from six months it just wasn't a problem for the rest of the company.

37signals was named after the 37 interesting signals which have been found as part of the SETI project.

37signals has a plan of not planning ahead. They don't add new processes or tools into their team because they think they will need them later, they wait until the need is real. Many company don't plan ahead, but few of them do it on purpose.

They also creates the tools they use. They run their company using the tools they sell. This means each tool they use was made to address a specific problem they had. They have never bought a big collaboration tool like Microsoft SharePoint just because they needed a small part.

The Tools

When 37signals was only three people they didn't need any specialized collaboration tools. With a group that small you can do well with just email and IM. As their company they needed a tool that supported collaboration with the whole team instead of between just two people. They needed IRC, but with a twist.

Campfire

Campfire is an easy-to-use browser-based community chat client. It works like IRC. The idea is very simple: a browser window where you can type a message and have it sent to everyone else. However, this simple idea causes a fundamental shift in the way teams work together.

The hallway conversation is the biggest loss with a distributed team. Communication needs to be easy and arranging a time for everyone to have a conference call just isn't easy. Community chat programs like Campfire and IRC takes the place of hallway conversations. Very similarly to the way that Mozilla runs on IRC, 37signals runs on Campfire. It has become a critical part of how the team operates.

Journal

Campfire solved one fundamental problem of a 37signals team member: *how can I ask the rest of my team simple questions without being disruptive*? When the team grew a little larger the team members had another problem: *how can I tell what other people on the team are doing*?

There is a fallacy that working in the same office lets everyone know what everyone else is working on. The idea is that you can absorb information about someone else's status through osmosis. Most engineers have a good understanding of what their close peers are working on, a vague understanding of what the rest of their team is doing, and no idea about anyone else in the company.

Being a distributed team doesn't make it more difficult to know what the rest of your team is doing, but it does make that lack of knowledge more obvious. Many teams solve this problem with status meetings or weekly status report emails. When 37signals had this problem they created a new tool called Journal.

Journal is once again very simple. It has fields where you talk about what you are working on and what your current status is and the rest of your team can see them. Journal goes beyond setting a status message works more like a micro-blog for just your team. If Jason wants to know what David spent his day doing, Journal gives him a pretty good idea. If David wants to know if Jason is available to talk then Journal will tell him.

Master Plate

With Campfire and Journal the 37signals team had a tool to support team discussions and a place to see what the rest of the team was working on. Over time the team added more products and those products had more features. It became more difficult for each person to answer the question *what should I be working on next*?

What should I be working on next is the fundamental question that almost every software project methodology —from waterfall to scrum— tries to solve. Most of them solve it with long meetings and complicated tracking tools. 37signals solved this problem with Master Plate and a different business model.

Master Plate has a very simple UI (notice the pattern here)? It is a web-based tool where you can see what everyone should be working on. Every team member has three items on their Master Plate: what they are working on now, what they should be working on next, and what they should be working on after that. Those three items are about as far into the future as Jason and David like to plan, and that is where their business model makes a difference

37signals sells their software with subscription model. Their users pay a low monthly rate to access the tools of their choice. This business model keeps 37signals very close to their customers. Instead of making one big sale and leaving a company alone for a year, 37signals has the chance to interact with their customers constantly. This constant contact makes it very easy for them to prioritize their work. The three items on Master Plate are the three items their customers have been asking for most.

A Very Small, Very Loyal Team

37signals employs 16 people including Jason and David. During the nine years since it was founded only two people have left the company. Keeping a consistent team helps the company stay focused. David is adamant that keeping the team you have is "more efficient, easier, cheaper, better on all accounts."

Everyone would agree that low employee turnover is a good thing, but making it happen is difficult. Jason and David put a lot of time and effort into figuring out how to make their working environment so good that people just won't want to leave. Their basic principles are simple: be reasonable and flat.

Be Reasonable

Jason and David try to remove every piece of red tape from their company. One example is expense reports. Many companies have large and complicated processes for filing expense reports. In my career I have had to staple receipts in the proper format to an 8.5 X 11 sheet of paper, print out emails to prove that I paid for shareware, and complete an expense report training program and test before I could submit my expenses.

Jason and David solved this problem by eliminating expense reports. Every employee gets a credit card with no limits and the instructions, "spend it wisely." This demonstrates how much trust 37signals puts in each member of the team.

> David:
> "Tons of companies act in very irrational mistrusting ways when it comes to their employees. We take the opposite approach."

Jason and David run their company on the idea that they should be reasonable. They are reasonable in their processes —they remove any impediments to productivity that they can— and they expect employees to be reasonable in return by not taking advantage of them. If one employee does cause a problem they change the policy for that one person and not the entire team.

Stay Flat

Jason and David strive to make the 37signals team very flat and open. The way they manage their office space is a good example. The team is based in Chicago, and five team members live nearby. However, everyone works from home. David explains why:

> David:
> "You can't have a preferred flock. If everybody in Chicago was meeting up every day in the office and talking about stuff and making decisions it would feel very 'second rank' not to be in Chicago. We don't want that atmosphere at all."

Jason and David have found the best way to maintain an open team and avoid interruptions during the workday is to work predominantly from home. They keep their office space for customer meetings and as a place for the team to meet together if they choose to. They are focus on "letting people work where they feel most comfortable."

Simple And Easy To Use

You may have read this section about 37signals and said to yourself, "that's great, but I'm not a manager." You probably don't have the power to change the expense reporting mechanism of your company or make everyone work from home just because you do. However, that shouldn't stop you from influencing the way your team works.

Do what you can to make your team easy to use. Remove unnecessary steps from your team processes. Encourage your team to be flat. You can't make people stop coming to the office, but ask them to try dialing into conference calls from separate rooms.

As a rule, 37signals doesn't add anything to their company until they absolutely need it. Jason and David are responsible for this, but so is the rest of their team. Many companies, teams, and software products could benefit from this practice.

Understanding and Resolving Problems

The cost of working remotely is that problems can sneak up on you. In the office you have many opportunities to communicate with your coworkers. Being remote makes it difficult to tell if someone is unhappy with your performance, frustrated with the difficulty of communicating with you remotely, or just having a bad day.

Learn how to read the warning signs and see small problems before they become big ones. Problems are always easier to solve earlier rather than later.

Signs of Trouble

You don't have the normal cues to find interpersonal issues so you need to use new ones. Every form of communication has additional, often unintentional, information. You need to read this information to look for signs of trouble.

There are two categories of remote warning signs: when changes happen in established patterns and when team members in the office are treated differently than those who work remotely. Every time an established pattern in your team changes it is a sign you should pay a little more attention to your team. If your chatty team suddenly stops talking it is a warning sign. If your normally quiet team becomes chatty it is a warning sign. As you get to know your team you will establish a baseline and learn to recognize changes.

If your team includes some members who are remote and others who are not, these two groups will be treated differently. Being in the same location as your manager allows them easier communication and that leads to differences. Differences that impact your work are a problem.

When you do find a problem, don't panic. Many problems can be solved with a short conversation. If your problem is with someone specific then talk to them and see if you can resolve it. Even more serious problems are usually solved with a little time and patience.

Big or little, all problems are easier to solve if you find them early. Make it part of your normal routine to check in with your team. Take an interest in what they are doing and see if they are having any trouble. Build trust ahead of time, and keep an eye out for signs of trouble within your team. Specific warning signs include:

You notice sudden changes in the level of communication. Most teams have a baseline level of communication. There are more discussions when projects are started and less chatter when you are in the middle of the implementation. If the amount of communication on your team changes suddenly then look for a reason. If you can't find one, then it is a sign of trouble.

Important decisions are made without you. If your coworkers are in the same location they will spend time talking without you. You can't prevent coworkers in the same location from having conversations without you. There are some discussions you will never be part of, but it is a warning sign when most decisions are being discussed and made exclusively by people in the office.

The solution is becoming a valuable part of the decision. When you work in the office people have to go out of their way to exclude you. You will come to the meetings and overhear conversations automatically. Work remotely means people have to go out of their way to include you. Make it worth their while. Make substantive contributions to the decision making process and they will make sure to include you. You don't have to be a superstar, just a clear net plus.

You are surprised by large personal events. If big personal events happen for your coworkers without your knowledge than that is a problem. Make sure to show an interest in your teammates. Ask them specific questions that show you know and care about their life and family.

Signs of trouble may not be actual problems. They may be indicative of a larger issue, or they may not. This makes them very difficult to judge. Take warning signs as information. Learn from them to better understand your team.

Beware of the Black Box

Many engineers see themselves as black boxes; specs go in, code comes out, and there is no information about your progress in between. The black box is a very exclusionary style of working with no room for collaboration. The problem is not enough information about your work process. After all, no engineer is an island.

Some engineers see themselves as the black box all the time, but others only enter this working mode when they are overwhelmed. Even if open communication is a part of your normal routine, you will naturally try to remove distractions and focus on your work when you are stressed. Removing distractions is a good idea, but you can't treat all interactions with your teammates as distractions. Look out for these specific warning signs that you are working in a black box style:

Nobody knows your status. If nobody knows the status of your work then you are in trouble. Somebody on your team should know what you are working on at all times. Even if you are having difficulties with your work, bad news is better than no news.

Nobody understands what you are working on. There should be at least one person on your team with a detailed understanding of what you are doing. Start each project by explaining your design and then keep your team updated throughout the process. If nobody understands what you are doing, nobody will understand how good it is.

You aren't asking questions. If you want people to take an interest in your work then you must show them the same courtesy. When you don't know what other people are doing, they probably don't know what you are doing.

You want to fix every bug before you show your work. It is better to show your team something broken than nothing at all. Show your team your work before it works.

Fixing Problems

"Be the change you want to see in the world. - Mahatma Gandhi

When you are remote you are physically removed from your team and you are physically removed from team problems. Those problems still affect you, but they are a lot easier to ignore. Don't use your remote status as an excuse to avoid problems.

Attack The Problem Head On, Then Back Off

If there is a problem within your team then you must attack it head on. Talk to the other members of your team. Be honest about the issues and get everything out in the open. Focus on solutions, not blame. This is easy to say, but it can be the most difficult part of the whole process.

Express your problem in clear objective terms that show how it is negatively impacting you and the rest of the team. Then back off. Don't push your coworkers to solve problems immediately. Most of the time just raising the issue is enough.

Blame Isn't Productive

Blame is a specter hovering around every problem. Ignore the urge to blame people for problems. Stay focused on solutions. When you talk about issues make it clear that you are speaking from your own point of view. You can't know what other people are feeling and it is insensitive to imply that you do. Use a lot of "I" statements to make it clear that you are talking about your feelings.

- I feel like I'm not being listened to.
- I'm worried about our schedule.
- I'm not getting enough information to understand the requirements.

Example

Talking about how to fix a problem is extraordinarily difficult. Let's look at a hypothetical example:

You have just suggested a new way of solving a problem and your coworker Joe gave you a flat out "no" You could call them on the phone and ask them directly "why don't you like my solution?"

This is very direct, but it is also very confrontational. This method implicitly accesses Joe of doing something wrong. It also assumes that Joe has looked at your solution, understood it, and then thoughtfully dismissed it. Maybe Joe was just busy and didn't get a chance to look at it, or maybe Joe has his own solution and doesn't want to think about another one. You just don't know. A better way to handle the situation is to listen more.

Start with an "I" statement: "I would like to know more about this problem. Can you help me?" Letting Joe talk first gives your more information about what he was thinking and makes it clear that you don't want to force your solution on him. Listen to his description and ask a lot of questions.

While the two of you are talking it will probably become clear which solution to the problem is better. Then comes the hardest part. If your solution isn't the best you have to let it go. It is difficult to abandon work you have done, but finding the better solution is more important than using your solution.

Don't Lose Your Cool

Blaming someone will not help you solve the problem. It is fine to get emotional, angry, or upset, just do it at home. Buy yourself a punching bag if you need to. Always be professional at work. Talk about what is going wrong and what you think needs to happen to make things go right. Do it clearly and calmly. It doesn't matter how you got into trouble, it matters how you are going to get out of it.

Solve Big Problems Face-To-Face

Any small problem can become a big one if you add a little time, a little indifference, or a hot temper. If you try to resolve a problem remotely and it doesn't work then you may have a big problem. When big problems happen it is time for a face-to-face meeting. A lot of these problems are communication problems and it can be difficult to resolve them over the phone. Show your team how important the issue is to you by travelling to resolve it.

Keep The Lines Of Communication Open

If you can't communicate then you can't solve anything. As difficult as it may be, just keep talking. Make sure you are available for the rest of the team and insist that they are available for you. When in doubt, ask questions. Ask a lot of questions. Ask about the weather. Ask about sports. Ask about anything. Just keep talking.

It Gets Easier With Time

The first problem you try to resolve will probably be the most difficult. As a team your group will learn to resolve problems and get better at it each time you do it. Be patient and keep working at it.

Fixing Remote Problems

Working remotely has its own pitfalls. If everyone else is in the office then remote problems will be seen as your responsibility. You may not be the cause, but you are the focal point. Here are some issues which are common in remote teams.

The Two-Tiered Team

The two-tiered team problem was the main issue for the Subversion team.

Having a two-tiered team was a problem the Subversion team had to solve early on. The core group had worked together in the past. They had a lot of history together and an established working relationship. They made all of the important decisions and had most of their discussions separate from the rest of the group. Left out members felt excluded from the team. It also meant that the team was missing out on valuable input.

The first step the Subversion team took to solve this problem was to understand it. They figured out what was happening, and saw that it was hurting the cohesion of their team. Once the team understood the issue they changed their work habits and the culture slowly changed.

They changed the way the team works. They stopped making decisions in person or on the phone. Discussions happened on the developers' mailing list and face-to-face discussion became secondary. The core team didn't stop talking or working together, but they accepted that nothing was official until it had been presented to the entire team.

> When there were originally only 4 or 5 developers on the project, we thought it was silly to have "trivial" dialog on the mailing list, so we mailed each other privately. Brian Behlendorf, CTO of CollabNet at the time, wisely yelled at us and forced us to do every communication on the public list. We thought his insistence on this was ridiculous, but as soon as we complied we started attracting some unbelievable volunteer developers to the project. (Thanks for setting us straight, Brian!) - Ben Collins-Sussman, Subversion

To flatten your team everyone needs to participate in discussions and make decisions, not just the core. If you are in a two-tiered team, you need to educate the rest of your team. Let them know what is going on, give them specific examples, and make sure to keep it positive. Don't tell them they are leaving you out; let them know that they are missing your important voice. Your team hired you because you are good at your job. When they don't include you they are losing out on your contributions.

The Power of Positive Visualization

When I am having trouble staying focused I take a moment, close my eyes, and imagine how happy my team will be with my amazing progress. I like to freeze them in my mind with a satisfied smile when they realize how great my work is. This simple vision gives me a specific goal and helps me stay focused.

You Aren't Getting Your Work Done

> Most offices are set up for interruption. They promote interruption by having everybody together all the time. And you can't actually get work done when you're interrupted or you're being interrupted. - Jason Fried, 37signals

Offices are designed to make it easy to disrupt people. This is why many telecommuters are more productive. However, you trade the distractions of the office for the distractions of home. Being remote puts the emphasis of being productive on you. Poor productivity will confirm your team's worst fears about letting you work remotely. Here are some common for not getting your work done and how you can fix them:

Goofing off

Goofing off is an important way to help you relax. It is crucial to build time into your working schedule to get away from your work. Get outside and take a walk or spend some time with your family. A little time away from the "office" can help you focus more on your work. However, if you aren't getting your work done then you have a big problem.

You can’t concentrate

Most office buildings are really boring, with passive color schemes and bland corporate art to remove distractions. When you work from home you are surrounded by your life. Many people find it helpful to create a simple office space free of distractions. Set up a very plain workspace. It will help you stay focused.

Too many distractions from home

Your friends and family will all understand the “home” part of working from home. They may not understand the “work” part. The physical separation of commuting to an office makes it clear when you are working. You need to consciously create this separation at home.

Hotelling

Hotelling is a recent office management strategy. Instead of assigned desks each employee uses a different workspace every day. This allows the office to support people who are only in the office part time. Services like Beta House in Cambridge, Massachusetts offer desks specifically for high tech professionals starting at $225 per month.

Working part-time in a shared office like this also helps you stay connected with other professionals in your area. One member of Beta House told me he got many business contacts and his current job through the people he met in a shared office.

Stay focused and productive

Your family, friends, hobbies, as well as the Internet, can all distract you from working. Don’t let them get in the way of being good at your job. Stay focused and practice a little discipline in your work day.

Close the door. The physical separation of a closed door is a powerful symbol. Close the door when you need to get work done.

Set expectations. Keep a steady work schedule and let people from home and work know when you will be working and when you won’t.

Separate communications. Your working communication and your personal communication should be separate. Use a separate phone line, email account, and IM username. Keeping your communications separate allows you to turn off outside distraction and still be connected to the office.

Create a log of your workday. If you aren’t getting your work done start by making a log of your day. Write down everything you do as you do it. For an added benefit you can classify your log entries as work-related or not. For many people, writing down how they spend their time is an eye-opener that gets them back on track.

Buy a personal computer. Keep your work computer for work and nothing else and use your personal computer for Internet browsing and games. Don’t turn on your personal computer until you are done working.

Remove distractions from your work computer. Uninstall the entertainment software from your work computer. Take out all the games and move your nonessential bookmarks to your personal computer. Remove Flash, Quicktime, and iTunes.

Don't work from home. If you home is too distracting then find somewhere else. Many coffee shops and public libraries provide wireless Internet access.

Work in an office. Most large cities have buildings offering office hotelling. If you need a professional environment to get your work done then pay the extra money and work in an office.

Your Teammates Are Resentful Of Your Remote Status

If you are working remotely in a team where most people are in the office your coworkers may get a little jealous. If so, they probably won't tell you. You need to give them a little reassurance that you are working hard.

Time Zone Troubles

You absolutely must share at least three working hours with your team every day. That means three hours when you and your team can communicate in real time. You won't talk for three hours every day, but you need the window of availability. Creating an open team requires real-time communication. Do whatever it takes to find time for it, even if it means working early in the morning or late at night.

- Get your work done. Getting your work done and producing clear results is the best way to put jealous feelings to rest. Make it abundantly clear that you are working hard and being productive.
- Be available. Let the team know that being remote does not mean being unreachable. Make sure you are available when your team needs you. Make them feel comfortable about calling you at home.
- Don't rub it in. Being remote has many advantages. Don't brag about them. It is even a good idea to complain a little.

A Coworker In The Office Is Taking Advantage Of Your Remote Status

Nobody likes it, but from time to time you will be involved in a workplace power struggle. It may be unclear what you are arguing about, but somehow you find yourself in an adversarial relationship with one of your teammates. If you are remote and the other person is not, you are at a big disadvantage. Your coworker in the office can get their way a lot more easily than you can.

If you find yourself in this situation, start by taking a deep breath. It is normal to take these situations personally, but you may be blowing things out of proportion. Give the other person the benefit of the doubt. They may not even realize you are having a problem. Talk to them about the situation. There is a good chance that you can resolve the problem with a short conversation.

This is another place where blame can be a factor. Focus on presenting your situation as a problem you need help with, and make sure to use a lot of "I" statements:

- I feel like I'm not part of the decision process.
- I'm not happy with my limited role on the team

If this doesn't resolve the issue then you need to handle the situation just as you would if you were in the office. You may want to confront the other person directly or involve your manager. Focus on concrete examples of the problems you have had. One big advantage you have when you are remote is that most communication is archived. If you can show an email or an IM chat log which backs up your position it can go a long way with your manager.

Whatever you do, don't lose your connection with the rest of the team. Have a confrontation with a coworker can be very frustrating, but you still need to work with everyone else. Losing your connection with the rest of the team means losing their support.

You Aren't Getting Promoted

Not getting promoted when you feel like you deserve it is a very difficult position. The problem is so vague. Did you get passed over because you aren't good enough at your job, didn't impress the right people, aren't visible enough in the company, because you work remotely, or was it some combination of these factors? In most situations you will never know.

Getting promoted as an individual contributor

When you are trying to get promoted there is a big difference between being promotion to a more senior postion and being promoted to a management position. If you were hired as a junior team member and want to be promoted to a senior one the solution is simple: do the job of a senior contributor and make sure the rest of your team knows it.

Doing the job is up to you and it doesn't change because you work remotely. Look at the senior members of your team. Are they taking on more responsibility than you? Are they more proactive about solving problems or designing larger solutions? Use them as a model and try to work like them.

Making sure the rest of your team knows what a good job you are doing is a two step process: do an good job and brag about it. Bragging has a bad reputation. Has anyone ever told you to be humble and wait for other people to notice you? It is really bad advice.

Bragging is important in the office too, but you probably do it without thinking about it. You talk about what you are working on at the lunch table or in the hallways. You call friends over and show them something you just figured out how to do. You need to do the same thing remotely. Go out of your way to show your team what you are working on and why you feel good about your work. Excitement is contagious.

Getting promoted to management

You are already doing the job of an individual contributor so getting promoted to a more senior one is relatively simple. Your current performance already proves that you can be an effective telecommuter. Getting promoted to a remote manager is much more difficult.

Many managers worry that telecommuters are more difficult to manage. Letting you manage remotely applies this concern to everyone you might manage. Apply a *try before you buy solution*. Ask for an opportunity to be a team leader. Take on a small project with a short time frame. Make it clear that you understand it is just a trial and there won't be any hard feelings if it doesn't work out.

The key is to show that you can manage people remotely in a situation with very little risk. Make the project work well and you will resolve a lot of the doubts about your ability to manage remotely.

Your Manager Wants You Back In The Office Too Often

Almost all telecommuters need to go back to the home office some of the time. How often you go back depends on the needs of your team and how close you are. Most companies bring everyone back at least once per year. Mozilla has all of their engineers return to an office once per quarter for an event they call work week. Meeting face-to-face with the rest of your team is an essential part of making your team strong.

However, some companies take this way too far. The most extreme example of this I have seen was an employee who boarded a plane every two weeks to fly back to the main office. Returning back to the home office this often is not only unnecessary, it is detrimental. When you return to the office this often you are in danger of running into the half and half problem.

The half and half problem is common for workers who work from home two or three days a week. They are in the office often enough that they don't feel the need to communicate remotely. They cut off all communication during the days they work from home. The person working remotely likes this setup because they feel productive during the days away from the office and connected with their team during the days in the office.

The problem with this approach is the way it affects the rest of the team. Managers are often left with the feeling that people don't do work when they are at home since they aren't communicating. This causes them to reserve the "perk" of working a couple of days a week from home for the top performers. They see it as letting them work a three day week because they do such great work.

Not communicating during the days at home also means the remote worker is not available to the rest of their team. They need to reschedule meetings and can't ask questions when the you are working from home. A clear sign that working from home part-time is negatively impacting the team is hearing, "we can't schedule the meeting for Thursday, Alice isn't in the the office on Thursdays."

So what does this all have to do with coming into the office twice a month? Coming back to the office every two weeks will cause the same behavior as working from home two days a week, but a lot worse. When you go to the office so often you will be tempted to wait until them to communicate. Even worse than that, the rest of your team will wait to communicate with you. Working remotely is much more likely to work if you stay away from the office. Returning back to the office more than four to five times a year is a real problem.

The solution to this problem is simple: stay away from the home office so you will communicate all the time and not just when you are in the office. However, it can be difficult to convince your manager. On the surface coming to the office more often sounds like a good thing. Tell them you don't want to wait until you get back to the office to do good work and you don't want your team to wait until they work with you. Try it out for two months, make working remotely work, and you will remove any reservations your manager has.

Virtual Problems With Real Solutions

When you are remote from your team it can feel like you are remote from their problems. The people in the office might be having a tough time, but you can just keep your head down and get your work done. Nothing could be farther from the truth. If you avoid problems because you are remote then your team will leave you alone until they call you to say goodbye.

Problems arise on all teams. Some of those problems happen because you are remote, some are exacerbated by the fact, and others are completely unrelated. If you take anything away from this chapter it should be that being part of your team means being part of the problems and part of the solutions. You are remote on your team, but you are very real. Your problems are real as well and they require real solutions.

All problems are stressful and you often can take advantage of your remote status to help you stay calm when everyone else is panicking. Drink a warm cup of tea, read a book, or take a relaxing walk. Release the tensions and stresses of the office for a little while and then come back to work and fix the problem. Assigning blames is easier than fixing problems, but far less productive. Address your problems in a straightforward manner when they arise and you'll succeed when everyone else is in a panic.

SECTION 4

BALANCE

Balancing Work Life and Home Life When You Live In The Office

I just can't stop working. I'm serious. It takes real effort for me to stop working. I love my wife and daughter, I love my friends, and I love my hobbies, but its always been difficult for me to stop working and spend time with them. Work feels so immediate.

I don't feel overwhelmed at work, it just feels like there is always something useful I could be doing. I like feeling motivated and it makes me feel good about myself to be good at what I do. In short, creating a good work/life balance is really hard for me. And I'm not the only one.

Almost everyone has trouble finding a work/life balance that works well for them. It doesn't matter if you tend to work too much or too little, the problem is the same: how do you work enough that you are successful at your job while still finding the time to have a life outside of work. It is a question we all face. And the bad news is that working from home makes it more difficult to find the answer.

Creating A Good Work/Life Balance

The best part about working from home is that you don't have to leave your house and the worst part about working from home is that you never leave the office. You have to make your coworkers feel like you are available when they need you, but you can't do that at the expense of your happiness or your family. Having a life outside of work is not only enjoyable, it is necessary. You have to spend some time unwinding and pursuing goals outside of work. If you don't find time to relax, you will start to burn out and your work will suffer.

Keep Your Promises

Do your best to be available when you say you will be. Be available to your coworkers when it is time to work, and be available to your family and yourself when it is time to stop working. Coworkers will always contact you outside of your scheduled working time. This is especially true if you are working with people in other time zones. Nobody means to be impolite; they just forget about the time difference.

Make sure you display your status somewhere your team can see it. IM and IRC are good tools for reporting your status. When you are away let your team know that you are away. When people send you messages you let them wait a little while for a response. Screen your calls. Keep a separate phone line for work, or use caller ID. Remember that you can always call them back later.

Flexibility vs. Accountability

Flexible hours is a big part of what makes working from home great. You can get your work done on a schedule that works for you. The cost of flexibility is a lack of collaboration. If nobody knows when you will be working then they don't know when to contact you.

Establish business hours. They don't have to be eight hours every day and they don't have to be planned out to the last minute. The goal is to set consistent expectations about when your team can contact you and when they can't. Make it easy for your coworkers; make it clear that they aren't disturbing you. Setting a schedule during the day can be very helpful.

Example schedule:

9AM to noon	Expect a quick response
noon to 1PM	Out to lunch
1PM to 4PM	Expect a quick response
4PM to 8PM	I'll respond, but it might not be very fast
8PM to 9AM	Don't contact me unless there is a really big emergency

Your schedule is an informal guide, not a rigid template. The key is consistency and setting expectations. Make it easy for your team to remember your schedule. Remembering that you work in the mornings is much easier than having to look up your current status.

Getting Away For A Little While

The schedule helps your team, but it also helps you. When you start working from home you may never stop working. You start early in the morning, close the door, and spend the day without coworkers walking over to ask you questions. It feels great to get so much work done and you work late into the evening most days. It is fine to keep this schedule occasionally, but it isn't sustainable. Use your schedule to help you take breaks. It is important to make time to get away from work every day.

Move

Move. Don't sit in front of the computer all day. Get up and move around. Take a walk, go out for lunch, sit in the park for a little while. Make exercise and basic movement a part of every day. Find a little time to stretch.

Get away for an hour every day and get up and move at least once every hour. Take a quick walk around your house or just stand up and stretch.

Working From Home Can Also Mean Working Near Home

Working from home doesn't mean you always have to work from home. Take your laptop to a coffee shop or local library. Work from the park on a nice day. Taking time away from your office space can help you relax and stay more productive.

Get distracted

In the office there is a pressure to always look busy. Most projects have busy times and slow times, but looking like you don't have enough work can be a liabiliy. Many offices have pool tables, ping-pong, and other games, but they also have a feeling that those games are for lunch time or after work. This is unfortunate since distractions help you be more productive.

You never need to look busy when you work remotely. Your boss won't walk past your cubicle and see you doing some idle Internet browsing. Take a break. Get out of your home office and get distracted. Run errands, put a basketball hoop outside your house, or read a book. Planning time for distractions helps you stay focused during work time.

Julie C.'s Flexible Schedule

Julie C. works in Kansas in the central United States. She is a program manager for a large global corporation and she keeps a more flexible schedule to make sure she can take time away from work.

"I tend to work the hours that are required for my job, and a lot of the people I work with are on the East coast of the US or in Europe or Asia. My day often starts early--it's not unusual to have 6:30 or 7:00 AM calls--and I often come back to work after the kids are in bed. I work some hours on the weekends, too. That sounds horrible, but I feel very compensated by the fact that if I'm not busy, I can go get my hair cut during the day, or run errands, or whatever. My work-life balance is pretty excellent."

Taking Vacation

Vacations from work are important, but taking a vacation when you are remote can be scary. What if I go away and nobody notices? What if someone tries to contact me and doesn't realize I am away? The solution is to plan your vacation ahead of time and notify everyone you can think of about your upcoming break. If your company uses a shared calendar then list your vacation time. Make it easy for people to know that you are away. Once again, it is all about setting expectations.

Finding a decent work/life balance is the only way to make working from home a long-term solution. Don't wait for that big project to end. Make spending time away from work part of every day.

Staying Focused

You need to find time to get your work done too. It sounds funny, but your coworkers may interrupt you more when you work from home. In the office someone can see that you are busy or that your door is closed. If someone calls you they just assume you will stop what you are doing and talk to them.

This isn't an issue on every team. If your team doesn't communicate well you will have problems, but time to get your work done won't be one of them. However, if your team, or just a couple of members of your team, are especially chatty then you will have a problem finding the time to finish your tasks.

Many cubicle dwellers solve this problem with a virtual door: a simple sign outside their cubicle which reads "**THE DOOR IS CLOSED**." This makes it easy for everyone to tell that the person inside is busy and shouldn't be bothered unless it is really important.

A closed door is the opposite of good communication, but this is another place where balance is required. You need to be available for your team so you can collaborate, but you can't chat on IM and finish your tasks at the same time. Getting interrupted by a coworker is a good thing —it means they are asking for your input— but it comes at a cost.

Every time someone asks you a question you have to stop working to answer it. Every time an instant message pops up on your screen it disrupts your train of thought. IM, IRC, conference calls, and emails can cause your productivity to die the death of a thousand cuts. When it is time to work, close the door by setting your status as "busy".

How To Stay Focused... Without Being A Jerk

Avoiding interruptions is easy if you cut all forms of communication. Unplug the phone, quit your email and IM clients, and remove any other way your team can contact you. You will work in a blissfully uninterrupted black box. You will also prevent communication and collaboration with the rest of your team. Nobody wants to work with a black box. Shut yourself off like this and you will confirm everyone's worst fears about working with remote engineers.

Make it clear to your team that you will be more interactive during some parts of the day. Use your schedule and keep it consistent. Setting aside one part of the day for questions and the other part for more focused work is a good model. When you are in a more focused working mode set your status so the team can tell what is going on.

The schedule won't stop everyone from interrupting your focused work time. Stand up to the people interrupting you. Tell them they will have to wait, but be polite. Make it clear that you are busy right now and tell them when you will get back to them. If you can give a quick description of what you are doing that is even better.

If all else fails

If you have set up your schedule, set your IM status to busy, and politely told people you need time to work, and they are still bothering you, you have my permission to disconnect. When you are running up against a deadline, it is OK to unplug the phone and turn off the IM so you can get your work done. In the rare cases that you do unplug to get your work done, remember:

Set expectations. Tell people that you are going to be away and tell them when you will be back online.

Leave one way to contact you. Even if your time is really scarce, you can't disconnect completely. Give your team one way to contact you. Letting them know you will check email a couple of times a day is fine.

Don't stay away too long. Disconnecting to finish an important project is OK, but don't make it a habit. Don't stay away any longer than you absolutely have to.

Socialization

Don't let working from home make you a hermit. Being in the office is a chance to socialize with your coworkers. It is also a reason for paying attention to the basics like personal hygiene and getting dressed in the morning.

Socialization is a skill like any other and you have to stay in practice. You may not be the most outgoing person, but everyone needs to spend some time with other people. Make commitments. Sign up for new classes, do some volunteer work, or take up a new sport.

Vulnerability To Depression

If you don't see other people you will get lonely and depressed. The correlation between isolation and depression is well known, and working remotely can quickly lead to depression. The risk of depression is highest during the second half of your first year working remotely. This is the time when you will settle into a routine and become less excited by all of your newfound freedom.

Make a plan for this point. Have a support system in place, through your friends and family, so you don't get into trouble. Give yourself structured tasks outside of the house. Making the time to get away from work and socialize will make you happier and more productive.

Isolation

Working alone far away from your teammates can cause feelings of isolation. This can be especially true when you are trying to solve difficult problems. Don't keep quiet about the problems you are facing. If you are having problems with your work then share them with the rest of your team.

> Karl Fogel:
> "If you are working on something hard or you feel daunted the first hing to do is to go get into a live chat room and talk to someone about it or go get on the mailing list. Psychologically it is really important to not let yourself feel alone. Don't sit there spinning on the problem. Let people know what you are working on. Let them know what's got you stopped right now. And very very often, surprisingly often, somebody knows the solution. Even when they don't, you feel less lonely after you've done it."

The Home Office

Working remotely often means working from home. The space where you work has a large impact on how you work. When you are in an office surrounded by other people working, it helps you work. Your home will never feel the same as your office.

Home Office Basics

When you set up your home office there are a few necessities and a few more optional items:

Laptop or desktop? Your company will almost certainly provide you with a computer, but they might give you a choice or you might want to use your own. Which kind your get is a question of your preference and your budget. Desktops are more powerful and cheaper, but leave you stuck in one place. Just the option to work from different parts of my home was enough for me to get a laptop.

Desk or chair? A desk is optional. Some people like to work in a comfortable chair with a laptop on their lap. If you do want a desk you don't need anything fancy, something basic from your local office supply store will do.

A chair. You have to sit somewhere.

A door. You don't always have to close the door, but you need the option.

Internet connection. How fast an Internet connection you need depends on the systems you are using. If you use Subversion or Git with a web-base bug tracking system like Bugzilla then a DSL link or cable modem is fine. If you use a heavy-weight version control system like IBM Rational ClearCase or need direct database access you might want to upgrade to a fiber optic line.

A phone. We've talked about the importance of high quality phones already. Get one.

Wireless connection? Wireless Internet is optional, but very convenient and cheap. Many wireless solutions sell for under $100.

Printer. Inkjet is good enough, but you need something. It might gather dust for a while, but you will be happy to have it when you need it.

External monitor? Totally up to you if you have a laptop. Monitors are getting better and cheaper every day, but so are laptop LCD screens. If you do get a monitor then you should also get a desk. Holding the monitor in front of you all day is very tiring.

External keyboard and mouse? These are optional if you have a laptop. If you have a tendency to repetitive stress injuries then an ergonomic keyboard and mouse are a good idea. I keep them handy, but I don't use them all the time.

Fax machine? I never thought I would need it, but my combination printer, copier, scanner, fax machine has been one of the best investments I ever made. Mostly I just fax for expense reports and patent applications, but it is really nice to have it when I need it.

What will your company pay for?

Companies vary widely on what parts of a home office they will pay for. I've never heard of a company that wants you to write software and doesn't provide a computer, but after that there is no standard.

If your company will pay for more than a computer the next item is your Internet and phone connections. I have also heard anecdotal evidence of companies giving a budget for chairs, desks, and printers. Many companies don't have firm policies on these expenses so ask your manager what they will sign off on.

Decor

Your home office decor should be functional and comfortable. It also needs to fit your working style. Workaholics should leave some distractions around the office to remind them to relax and procrastinators should remove all distractions to help them stay focused. Figure out your work style and then organize your space to support a productive balance.

Separate Communications

Along with separate space, you need separate communications. Make sure the lines of communication you depend on are always available. Get a separate phone line and don't do your work on the family computer.

Being Professional From Home

Your home is a more casual environment than the office, but you still need to remain professional. Review all of the policies of your company and follow them at home as well as you would follow them in the office.

Propriety

The Internet is full of content which is not appropriate for the workplace. Viewing pornographic or obscene material in the workplace is often tantamount to sexual harassment. It also just isn't a very nice thing to do. When you are working from home you may be tempted to view websites and other material you would never think of looking at in the office.

Using the computer provided by your company, or accessing the Internet through your company VPN, to look at inappropriate material in your home is seen by many companies as the same as doing it in your office. There are hundreds of stories out there of people getting fired for the history in their browser. It is best to err well on the side of caution. Your work computer should be used only for tasks directly related to your work. If you want to spend time accessing extracurricular material then you should purchase a personal computer.

Intellectual Property

Most companies have you sign a contract including a non-competition clause. This normally means that you can't work for other computer companies without permission. This doesn't change if you are working remotely. Nobody is watching you work, but you still need to let your company know what you are working on.

Be careful about intellectual property issues. Some companies take the stance that they own any code your write while working for them. Others are more permissive. You need to contact your company for permission before writing any software you don't want them to own. That includes open source software.

Your Family

At home, there is nobody to watch you work. This is one of the best parts about working remotely. The flexible working hours allow you to take breaks. They also let you distract yourself with other tasks that, while they may be important, prevent you from getting your work done.

The key to being productive when you work from home is separation. Set aside specific time to work and not work. Separate yourself physically by having a separate office space and closing the door. Separate yourself emotionally by setting specific times to work and using that time to get your work done.

Get Your Family On Board With Working At Home

If you live with your family, then you can't work from home successfully without their support. Have a discussion with your family about exactly what you need and how you are all going to make it work. Ideally you would have this conversation before you start working remotely, but it can happen any time you are having trouble.

Working At Home With Small Children

If you have children who are too young for school then working from home can cause some very conflicting emotions. You probably started working from home because you wanted to spend more time with you family. Nobody wants to miss their kids growing up. However, you have to separate from your family enough to get your work done.

Physical separation Small children do not understand boundaries, schedules, or delayed gratification. The only way to coexist with a small child and get any work done is to physically separate yourself. Close the door. The door prevents your child from entering. It also prevents your child from seeing you. For most kids, once they see you it is all over.

Support You can't care for your children and dependably get your work done at the same time. Get some support to help you with childcare. You will also want to make time to see your children. The steady stream of work can cause you to work longer hours than you expected. This results in no time with your children and frayed nerves for the primary caregiver.

Robin M.'s Story

> Robin M. is an IT project manager at a Fortune 500 company. She has worked for the company for five years and has worked remotely for almost the entire time. She has a young daughter and is expecting a son. Robin says:
>
> "The company I'm with really supports IT personnel working virtually. I have been working from home 1-2 days a week since I started there, but when I became pregnant with my daughter I went virtual full-time. Being able to work from home was a real blessing when I was pregnant and nursing. After I weaned my daughter, I was promoted and began working from the office most days of the week; however when I became pregnant with my son, I decided to work from home again. Now I'm expecting my son in a few months and plan to work from home until he is weaned. After that, I'll reevaluate the situation. Having the option of working from home makes pregnancy and breastfeeding much easier in many ways - and allows me to have the ability to continue to focus on my work. I'm grateful to have the option."

Robin says the most difficult part of working from home with her young daughter was getting the proper support from her spouse.

> "Really, training her father is a harder thing - getting him to think about keeping her noise level down outside my office door and reminding her not to go in there has been the real key."

Working from home as a single parent

Working from home and providing full-time childcare is just as impossible as working in the office and providing full-time childcare. If you are a single parent trying to work from home you need some support. Relatives, friends, or professionals; someone has to be helping you. Trying to watch a toddler while attending a meeting is untenable.

Working At Home With Older Children

When your children are a little older you can take advantage of their time in school to get your work done. Get your kids ready for school, work while they are in school, and then take a break when school ends. You can always finish up the work you have left over while they do their homework. As long as you have help this can be a fine arrangement.

Completing The Balancing Act

Working from home is a balancing act. You will be pulled in many directions. You need to stay in contact with your team, but you want to get your work done. You need to stay focused on work, but you have to find time to take a break. You want to spend more time with your family, but you need to shut them out in order to be productive.

This balancing act can be maddening at times and there is no one right answer. What works for you today may not work for you tomorrow. Make it a constant part of your life to reevaluate your working situation. Do you need to change your hours? Are you getting your work done and still spending time away from work?

After that, just relax and do your best. Working from home gives you a lot of flexibility and you will find a solution that works for you.

Summary

Keep your promises to your coworkers to get your work done and yourself to spend time away from work.

Make a schedule that works for you.

Getting away from work will help you stay focused. Plan some distractions and don't forget to move.

Staying focused takes good communication. Let people know when you don't want to be disturbed.

Don't get isolated. Stay connected with your team about work, and stay connected outside of your team with activities where you can meet people face-to-face.

Give your home office some thought. A good space to work will help you stay productive.

Stay professional, even though you may be working in your pajamas.

Plan your work with your family. Make it clear when you will be working and how you will maintain separation.

Conclusion

At the end of 2007 I was laid off. I fell victim to the dotted line problem. For the previous six months I had been assigned to one manager but working for another. This is always a precarious position when downsizing happens. Your old manager is still paying you but your new manager is getting all the benefits of your work.

The company gave me 60 days notice to finish my project; I was getting ready to release a new subsystem and they wanted me to finish it. I tried to focus on my work, but I was also thinking about ways to stay with the company. Could I get them to increase the budget somewhere? Was it possible to move to a different group? Could I just make them change their minds?

It was tough because the downsizing was across the entire division and nobody had the resources to hire new people. As my last 60 days turned into my last 30 I became convinced I would have to leave. I worked on my resume, started a new professional blog (which I should have had already), and began looking at my professional network to see who could help me find my next job.

I didn't realize it, but during this time my dotted line teammates were fighting for me. They were talking to anyone and everyone they could think of; up to the highest levels of the company. They spent time, effort, and political capital trying to find a place for me to stay in the company.

With just two days to spare they succeeded. They convinced the company to make the dotted line solid and I joined their team. I was the only person from that round of layoffs to stay in the company. Those guys fought to make a place for me. They saved my job. And they had never met me.

My new team was located in a different part of the country and I had never met any of them face-to-face. We had established a bond while working together remotely. They knew I would be good for the team because of my interactions with them and not just the code I wrote.

I hadn't been an official part of their team before that, but I reached out to them. I joined all of their team meetings and I called them one-on-one to share ideas make connections. I made sure they could see my work and did everything I could to make it easy for them to give me an A. It saved my job.

The Principles of This Book

Communication is king.
If nobody knows what you did you might as well have not done it. The software you create is only useful when other people use it. Talk about what you are doing and why. Write about it too.

Everything is communication.
Communication doesn't just happen over the phone or in email. It is part of everything you do. Everything you share with your team is communication. Take advantage of every opportunity to share your ideas. Don't limit yourself to weekly status reports.

Present your own ideas.
Don't let being remote prevent you from presenting your ideas. Never let

someone in the office give your presentation just because of their location. Talk directly with your team, your managers, and your customers. Show them what you are doing and tell them why it is worthwhile. Be proud of your work and your ideas.

Emotional connections matter.
A team without trust will fail. Build that trust with emotional connections. Get to know the people you are working with. Know about their lives and their interests. Tell them about you. Create a bond with them. Let your coworkers know you as a real person instead of just a screen name.

Software is about people, not code.
Being a professional programmer is not all about programming. The way you solve a problem is not as important as choosing the right problem to solve. It doesn't matter if you succeed only to see your team fail. Work cooperatively with your team. Ask questions. Learn from your team and teach them. Create an open environment where you can find the right problems before you find the right solutions.

Fix problems early.
Problems come up in any team. Fix them early. Help the small problems stay small by staying on top of them. Look out for signs of trouble in your team and talk about them. Most little problems come from poor communication. Solve them by connecting with your team before they become big problems.

Find balance.
Finding balance in your life makes you better at your job. Don't let working from home mean you never stop working. Time away from work will help you stay focused and relaxed with your work.

Market yourself.
Market yourself before it is time to find a new job. Make connections and build your personal network. Work on projects other people can see. Give potential employers much more than your resume.

Use the lessons from this book. They'll make you a better programmer and a better team member.

Perfect Remote Communication

A way of gauging how much transmission of presence you are doing [is] by the kind of argument, and depth of the argument, that a communications system [can] handle.

- Alan Kay

When I started writing this book I made a list of all the communication technologies that someone might use in their office. It started with telephones and ended with carrier pigeons. A lot of technologies on the list didn't exist 10 years ago. 10 years from now there will be even more.

When new communication options become available, how do you determine if you should use them? Will they help you communicate or will they just get in the way?

The Dinner Table Criteria

The ultimate measure of a communication technology is dinner conversation with three or four friends. Everyone sits around the table in a quiet corner of someone's home. Wine is poured. Delicious smells waft from the kitchen. You relax in your chair, among friends, and just talk.

You could talk about anything around that table. The conversation could shift from a fine meal you had recently, to politics, to family, to relationships, to sex, to friends you've known in the past, to people you know today, to your hopes and dreams. With some good food and a little bit of time the discussion could go anywhere.

This is the power of a effective communication system. The bandwidth is enormous, but manageable. You can read the people at the table. You can tell how someone feels just by looking at them. You have the information you need to engage in complex communication.

We can use the dinner table to create three simple criteria with which to judge other communication systems.

Depth. There is no limit to the type of conversation you can have over a dinner table. Try discussing art on an IRC channel or aesthetics in an email and you will quickly see the limits of some communication technologies.

Ease of use. A dinner conversation is easy to use. There are no complex setup screens, configuration, or user interfaces to learn. Anyone can sit down and join in.

Bandwidth. The bandwidth of a dinner conversation is enormous. Not only can you hear everyone's voice with perfect accuracy, but you can see them with perfect clarity. Is someone tired? Are they well dressed? Do they look excited or bored? You can see them well enough to know the answer.

The Perfect Remote Communication System

Much of the way computer interfaces look today is the result of work by Dan Ingalls, Alan Kay, and others at Xerox PARC. I spoke with Dan and Alan about the perfect technology for remote communication and they both agreed: a wormhole, with a curtain over it for privacy.

Picture it. You are sitting at your computer and just to the side of your monitor there is a curtain that hangs in mid air. You open it and reach into another office to tap your teammate on the shoulder. They might be in the next room or across the world. They turn around and look you in the eye, face-to-face; rendered perfectly.

If we judge the wormhole with the dinner table criteria it scores extremely well. Is it deep? Yes! You could have any type of conversation there. Is it easy to use? Absolutely! There is nothing to setup or configure and everyone can understand a curtain. Does it have high bandwidth? Definitely! You can see the person in front of you as if you were in the same room.

Creating wormholes for personal communication will be beyond our abilities for a long time, but it is a useful way to think about remote communication. When you are presented with a new communication system you can use the same dinner table criteria —deep, easy, and high bandwidth— to judge it.

Apply the dinner table criteria to a video conference using Apple's *iChat*. You sit in front of your computer and see a two or three inch moving picture of the person on the other side. Is it deep? Absolutely not. The blocky facial representations and out of synch voices do not support complex conversations. Is it easy to use? Not at all. You need to understand complex setup instructions, firewall issues, and IP addresses. Does it have high bandwidth? No. You can't really tell if someone has shaved that morning, forget about seeing facial expressions.

Video conferencing scores low on all of the criteria. This explains why it isn't popular for important communication.

Some technologies do well by scoring high on two out of three of the criteria. Instant messenger is easy to use: just click someone's name and start typing. It is also reasonably high bandwidth, assuming you can type fast. It isn't very deep, but that doesn't matter as much for more trivial conversations. IM isn't very good for a deep conversation with a close friend, but it is fine for quickly catching up.

Not all high scoring technologies need to be visual. The telephone scores highly on all three criteria. You can have a deep conversation over the telephone. It responds quickly and does a good job of letting you communicate in real-time. Telephones are easy to use: just dial the right number. Telephones also have enormous bandwidth. You can't see what someone looks like, but a good quality phone provides high enough fidelity to pick out additional emotional information. You can tell if someone is happy, sad, or angry by listening to them talk on the phone. This explains why the telephone has been such an enduring communication tool.

The dinner table criteria also explains why bad cell phone connections are so frustrating. When your cell phone isn't working properly it has almost no depth because you have to repeat everything you say. If you have ever pressed your head against a glass window or stood outside to get better reception, you know that cell phones can be difficult to use. And the tinny voice quality of a bad cell phone connection makes the bandwidth low. All of your effort is spent just trying to understand what the other person is saying. You don't get any extra information.

When you consider a new communication technology think about how well it does compared to the dinner table. Many companies could have saved the money they spent in expensive video conferencing installations by looking at the depth, ease of use, and bandwidth of the technology.

People Featured In The One Minute Commute

This book was made possible by the many people who contributed with interviews and quotes. A simple acknowledgement didn't seem like enough so I wanted to give each person a little space of their own. Thank you very much to everyone for all of your help and support.

Ricardo Anguiano

Ricardo Anguiano joined CodeSourcery in 2001 and is the Director of Information Technology. He holds degrees in Computer Science from the University of California at Davis. In addition to creating software for his team he is also responsible for systems administration.

Jim Blandy

Jim Blandy is a founding member of the Subversion project and the creator of the initial design of the Subversion repository. He has also worked for CodeSourcery and Red Hat focusing on the GNU compiler and debugger. Jim currently works on Mozilla's ActionMonkey project where he is integrating a new JavaScript engine to help FireFox run faster.

Jim was one of the first people I interviewed for this book and he has more connections to the teams in this book than anyone else. Jim helped introduce me to CodeSourcery, Mozilla, and the rest of the Subversion team. He was also the first person to introduce me to the idea of the social currency of open source.

Jim was a strong influence on the Open Teams. He lives with his wife, children, and dogs in Portland, Oregon. Find out more about Jim on his website: http://www.red-bean.com/~jimb.

Chris Brogan

Chris Brogan is a ten-year veteran of using social media and technology to build digital relationships for businesses, organizations, and individuals. Chris speaks, blogs, writes articles, and makes media of all kinds at http://www.chrisbrogan.com, a blog in the top 20 of the Advertising Age Power150, and in the top 100 on Technorati. Chris's contributions strongly influenced the Market Yourself Into a Remote Job chapter.

Ben Collins-Sussman

Ben Collins-Sussman is a founding member of the Subversion project and is currently a technical lead for Google's Open Source Project Hosting. He is a programmer and a musician. He lives with his family in Chicago.

Find out more about Ben at his website: http://www.red-bean.com/sussman

Julia Cort

Julia Cort runs Julia Cort Recruiting, a firm that specializes in hiring remote compiler and tools developers. Find out more about Julia at http://www.juliacortrecruiting.com

Jono DiCarlo

Jono DiCarlo is the co-founder of a small Chicago company called Humanized and currently a member of the Mozilla Labs team where he is focusing on the Ubiquity and Weave projects. Jono graduated college at 17 with a degree in physics. He then earned an MS in Computer Science from the University of Chicago in 2005. He blogs about his current work and the need for better user interfaces in open source projects at http://jonoscript.wordpress.com.

Nancy Duarte

Nancy Duarte is the CEO of Duarte Design and author of *Slide:ology: The Art And Science Of Creating Great Presentations* She has worked with some of the greatest brands and thought leaders in the world such as Adobe, Apple, Al Gore, Cisco, Google, Hewlett Packard, Symantec and TiVO. Her firm is the largest design firm and a top woman-owned firm in the Silicon Valley.

You can learn more about Nancy's company at http://www.duarte.com and read her blog at http://www.slideology.com.

Paul Ekman

Paul Ekman is a psychologist and pioneer of the study of emotions and their relation to facial expression. He has been named one of the 100 most eminent psychologists of the 20th century by the *Review of General Psychology*. Dr. Ekman is the author of 10 popular and academic books. His latest book, *Emotional Awareness*, was cowritten with the Dalai Lama.

Dr. Ekman is the director of the Paul Ekman Group, LLC (PEG), a small company that produces training devices relevant to emotional skills, and is initiating new research relevant to national security and law enforcement. http://www.paulekman.com

Dr. Ekman has appeared on 48 Hours, Dateline, Good Morning America, 20/20, Larry King, Oprah, Johnny Carson and many other TV programs. He has also been featured on various public television programs such as News Hour with Jim Lehrer, and Bill Moyers' The Truth About Lying.

Daniel Einspanjer

Daniel Einspanjer is a metrics software engineer for Mozilla. He works remotely from Salem, NH. He blogs at http://blog.mozilla.com/data.

Brian Fitzpatrick

Brian Fitzpatrick has been an active open source contributor for over ten years. After years of writing small open source programs and bugfixes, he became a core Subversion developer in 2000, and then the lead developer of the cvs2svn utility. He was nominated as a member of the Apache Software Foundation in 2002 and spent two years as the ASF's VP of Public Relations. Brian has written numerous articles and given many presentations on a wide variety of subjects from version control to software development, including co-writing *Version Control with Subversion* and contributing chapters to *Unix in a Nutshell* and *Linux in a Nutshell*.

Find out more about Brian at his website: http://www.red-bean.com/fitz.

Karl Fogel

Karl Fogel is a founding member of the Subversion project and the author of *Producing Open Source Software: How to Run a Successful Free Software Project*. He is an expert in the social dynamics of open source projects and distributed teams. My interview with Karl greatly influenced the general philosophy of this book. Read more about Karl at his website: http://www.red-bean.com/kfogel

Bert Freudenberg

Bert Freudenberg is a freelance software engineer works for the Viewpoints Research Institute and ports Etoys to the One Laptop Per Child platform. He completed his Ph.D. thesis *Real-Time Stroke-based Halftoning* in 2004. He has won awards for his work on SIGGRAPH and Plopp. Bert lives in Magdeburg, Germany with his wife and four children.

Jason Fried

37signals doesn't have titles, but Jason Fried describes himself as "President, I guess." He is a founder of 37signals and a managing partner. Jason is also a contributor to the popular Signal vs. Noise blog http://www.37signals.com/svn.

Scott Hanselman

Scott is a Principal Program Manager at Microsoft, public speaker, and author of the popular blog Computer Zen. He manages a team of people working remotely around the United States and blogs often about working remotely.

David Heinmeier Hansson

David Heinmeier Hansson is a managing partner at 37signals, the creator of the Ruby on Rails framework, and a contributor to the popular Signal vs. Noise blog http://www.37signals.com/svn.

For his work on Rails, David won Best Hacker of the Year 2005 at OSCON from Google and O'Reilly. And in 2006, he accepted the Jolt award for product excellence for Rails 1.0. He has been featured on the cover of LinuxJournal and in the pages of Wired, Business 2.0, and Chicago Tribune as well as other publications.

Mark Horstman

Mark Horstman is co-founder of Manager Tools, a management consulting firm. Manager Tools has served firms such as Intel, Applied Materials, Cornell University, USAA, P&G, GE, the Federal government, and Microsoft. Manager Tools was recently the number one business podcast on Itunes in the world, has twice been named Best Business Podcast in the US, and was People's Choice Podcast of the Year (#1 in all categories) in 2008. Its podcasts are downloaded over 80,000 times each week.

Dan Ingalls

Formerly a Distinguished Engineer at Sun Microsystems Laboratories, Dan Ingalls is interested in dynamic languages, graphics and kernel software. He is Principal Investigator of the Lively Kernel project, a project to rethink web programming and the web itself.

Dan Ingalls is the principal architect of five generations of Smalltalk environments. He designed the byte-coded virtual machine that made Smalltalk practical in 1976. More recently, he conceived a Smalltalk written in itself and made portable and efficient by a Smalltalk-to-C translator, now known as the Squeak open-source Smalltalk.

Dan Ingalls did his first well known research at Xerox PARC where he began a lifelong association with Alan Kay. He has been a member of the Viewpoints Research Institute team off and on since the creation of the company.

Alan Kay

Alan Kay, President of Viewpoints Research Institute, Inc., is one of the earliest pioneers of object-oriented programming, personal computing, and graphical user interfaces. His contributions have been recognized with the Charles Stark Draper Prize of the National Academy of Engineering "for the vision, conception, and development of the first practical networked personal computers", the Alan. M. Turing Award from the Association of Computing Machinery "for pioneering many of the ideas at the root of contemporary object-oriented programming languages, leading the team that developed Smalltalk", and "for fundamental contributions to personal computing", and the Kyoto Prize from the Inamori Foundation "for creation of the concept of modern personal computing and contribution to its realization."

He has been a Xerox Fellow, Chief Scientist of Atari, Apple Fellow, Disney Fellow, and HP Senior Fellow. He is currently an Adjunct Professor of Computer Science at UCLA. In 2001 he founded Viewpoints Research Institute, a non-profit organization dedicated to children and learning.

Mark Mitchell

Mark Mitchell is the founder and Chief Sourcerer of CodeSourcery, Inc. He has worked on C/C++ software development tools since 1994. Mr. Mitchell has been the Free Software Foundation's GNU Compiler Collection (GCC) Release Manager and a member of the GCC Steering Committee since 2001. He holds degrees in computer science from Harvard and Stanford.

Mark, along with Alex Samuel and Jeffrey Oldham, is the author *Advanced Linux Programming*.

Catherine Moore

Catherine Moore is a Sourcerer in CodeSourcery's GNU Toolchains Group. She has been working on open-source software, including Binutils and GCC, for 14 years. She holds a BS from Boston College.

Jason Orendorff

Jason Orendorff got his job working on the JavaScript engine of the Firefox web browser by writing some really bad code. He blogs about Mozilla hacking and the FireFox SpiderMonkey JavaScript engine at http://blog.mozilla.com/jorendorff.

C. Michael Pilato

C. Michael Pilato is a core Subversion developer, co-author of *Version Control With Subversion*, and the primary maintainer of ViewVC. He works remotely from his home state of North Carolina as a senior software engineer on CollabNet's version control team, and has been an active open source developer for over seven years. Mike is a proud husband and father who loves traveling and spending quality time with his family. He also enjoys composing and performing music, and harbors not-so-secret fantasies of rock stardom. Until that all works out, though, he is content to spend his modicum of private time doing freelance web design, graphic design, audio and video production work. Mike has a degree in computer science and mathematics from the University of North Carolina at Charlotte. Read his blog at http://www.cmichaelpilato.com

Aza Raskin

Aza gave his first talk on user interface at age 10 at the local San Francisco chapter of SIGCHI and got hooked. At 17, he was talking and consulting internationally; at 19, he coauthored a physics textbook; at 21, he co-founded Humanized. Aza is currently the Head of User Experience at Mozilla Labs where he works on the Ubiquity project among others.

Aza enjoys playing the french horn and puttering in his lab when time permits. Read his blog at http://www.azarask.in.

John Resig

John Resig is a JavaScript Evangelist for the Mozilla Corporation and the author of the book *Pro JavaScript Techniques*. He's also the creator and lead developer of the jQuery JavaScript library.

Currently, John is located in Boston, MA. His blog at http://www.ejohn.org is an effective example of personal branding.

Garr Reynolds

Garr Reynolds is the author of Presentation Zen and the very popular blog http://www.presentationzen.com. He is a sought-after speaker and an expert on presentation techniques and self-branding. His personal website http://www.garrreynolds.com is an effective personal branding tool. His latest book is *Presentation Zen Design*.

Kim Rose

Kim Rose is the co-founder and Executive Director of Viewpoints Research Institute. In addition to overseeing all administrative and financial aspects of the non-profit organization, Kim is a media developer, media critic and cognitive scientist. She has been affiliated with Alan Kay and his research team since 1986 when she joined the "Vivarium Project" at Apple Computer.

Her publications include: "Squeak - Open Personal Computing and Multimedia" with Mark Guzdial, Prentice-Hall/Pearson, 2001, and "Powerful Ideas in the Classroom Using Squeak to Enhance Math and Science Learning" with B.J. Allen-Conn, Viewpoints Research Institute, Inc., 2003. "Powerful Ideas" has been translated into 5 languages.

Alex Rosen

Alex Rosen has been a software engineer for 14 years and has a BS, Computer Science and Engineering from MIT. He currently works at Endeca, a privately held company focused on enterprise information access software. He once got a job by warping Bill Gates.

Stefan Seefeld

Stefan Seefeld is a Sourcerer in CodeSourcery's High Performance Computing Group. He has contributed to a variety of open-source software projects over the past 12 years, including serving as a lead developer and maintainer for Fresco and Synopsis. He received his MS in Physics from Humboldt University in Berlin and has pursued graduate studies in Biophysics at Universite de Montreal.

Nathan Sidwell

Nathan Sidwell is the Director of the GNU Toolchains Group at CodeSourcery where he has worked since 2002. He has more than 10 years of experience on open-source projects, including GCC, GDB, and Binutils, and currently serves as a GCC C++ maintainer and VxWorks target maintainer for the FSF. He received his BSc and Ph.D. in Physics from the University of Bristol.

Benjamin Smedberg

Benjamin Smedberg works for the Mozilla Corporation as the coordinator and lead developer of the Mozilla XULRunner project since early 2005. He is the module owner of Mozilla's toolkit, embedding, and build system modules. Before switching careers, Benjamin was a professional choir director and organist who worked on Mozilla as a volunteer. http://benjamin.smedbergs.us/blog

Atul Varma

Atul Varma is the co-founder of a small Chicago software company called Humanized. He currently resides in San Francisco and works for Mozilla Labs on the Ubiquity project as well as others.

Atul graduated from Kenyon College with a BA in mathematics in 2001, and received his MS in Computer Science from the University of Chicago in 2004. Atul's career experiences are rather varied; in addition to a combined nine years worth of experience as a professional web developer, video game programmer, and usability consultant serving such high-profile clients as IBM Global Services and Samsung, Atul has also taught and tutored at public schools and literacy programs and worked for a variety of nonprofit organizations.

Atul blogs about Ubiquity, user experience design, and other topics on his blog http://www.toolness.com.

Chuck Wilsker

Chuck Wilsker is the President, CEO, and Co-founder of The Telework Coalition. TelCoa, is the nation's leading nonprofit telework education and advocacy organization addressing all forms of telework and telecommuting including virtual, mobile, and distributed work. The economic, environmental, and energy benefits as well as business continuity and the needs of rural, older and disabled workers are focus areas. Chuck is frequently quoted in the National media as an expert in his field.

Acknowledgements

I received a crazy amount of help and support while writing this book. Over 60 experts in various fields donated their time for interviews. Friends and family helped me review and edit and refine The One Minute Commute book. There were photographers, graphic designers, and lawyers. And they all deserve thanks.

37signals: Jason Fried and David Heinmeier Hansson

Code Sourcery: Ricardo Anguiano, Mark Mitchell, Catherine Moore, Stephan Seefeld, and Nathan Sidwell

Microsoft: Ade Miller and Scott Hanselman

Mozilla: Alexandra Cuccias, Jono Dicarlo, Blair McBride, Daniel Einspanjer, Jason Orendorff, Abimanyu Raja, Aza Raskin, John Resig, Ben Smedberg, and Atul Varma

Viewpoints Research Institute: Bert Freudenberg, Luke Gorrie, Dan Ingalls, Alan Kay, and Kim Rose

Subversion: Jim Blandy, Brian Fitzpatrick, Karl Fogel, Bert Huijben, Michael Pilato, Ben Collins Sussman

Sun Microsystems: Martin Balin, Janet Keeley, Martin Matula, and Heidi Pate

WikiMedia: Rob Church, Tim Starling, and Brion Vibber

Fatima Aydin, Jurgen Appelo, Sierra Black, Shelly Boose, Chris Brogan, Anita Bruzzese, Julia Cort, Esther Derby, Nancy Duarte, James Duncan Davidson , Paul Ekman, Justin Evans, Eric Fillion, Jesse Glick, Aaron Goodisman, Ted Grossbart, Helen Herold, Maggie Hessel, Mark Horstman, Jim Idleson, MaryFran Johnson, Steve Kasmouski, Karin Lundberg, Michael Meeks, Robin Murphy, Robb Perry, Joe Pusateri, Garr Reynolds, Alex Rosen, Johanna Rothman, Shimon Rura, Shev Rush, Rajesh Setty, David Shopper, Kerr Timbers, Rosely Traube, Shannon Vargo, Chuck Wilsker, Sarah Whedon, Sue Veitch, Mary Vogt

And a special thanks to my editors and reviwers Howard Melman, Susannah Pfalzer, Jonathan Simon, Daniel, Steinberg, Lyndon Washington, Ian Welsh, and Mary Vogt

Thanks to everyone else who helped with this book and isn't mentioned on this page.

❧ This book was designed for digital release by Zack Grossbart using Adobe InDesign CS4. The text face is Minion Pro designed by Robert Slimbach. The sans-serif text face is Scala Sans designed by Martin Majoor. The cover is set in Archer by Jonathan Hoefler and Tobias Frere-Jones.