Extracted from:

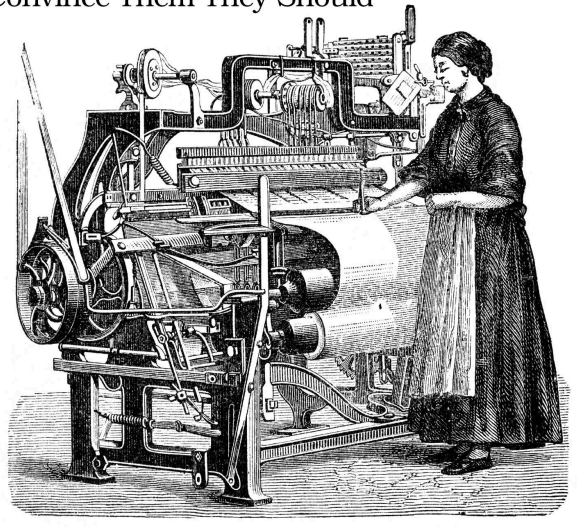
# Driving Technical Change

## Why People On Your Team Don't Act On Good Ideas, and How To Convince Them They Should

The Pragmatic Programmers

# Driving Technical Change

## Why People On Your Team Don't Act on Good Ideas, and How To Convince Them They Should



Terrence Ryan

*Edited by Jacquelyn Carter*

Chapter 2

# Defining the Problem

In this book we're going to be talking about selling professional development to skeptics. To get started with that, I think we need to lay out just what we are talking about:

- What do we mean by professional development?
- Who are these skeptics?
- Why do we need to sell it?

## 2.1 What Do We Mean by Professional Development?

Professional development includes any tool or technique that makes you more productive as a developer, your work less vulnerable to failure, or your code more understandable to your teammates. That covers a lot, so let's get more concrete.

Productivity would at first glance point to things such as automation and code generation that allow you to produce more code in a shorter amount of time. But it can also extend to languages. If you are able to create more functionality in fewer lines of code by using another language, that counts. I'll even go one controversial step further: you can make the argument that you can be more productive through your choice in operating systems.

As for vulnerability, the big thing that comes to mind here is source control. You can't feel safe today unless you are running some sort of source control. But it goes beyond that. Unit tests make you less vulnerable to bugs, as does UI testing. Code reviews can also make you

safer. Basically, anything that helps you sleep better at night or allows you to avoid the *getting hit by a bus* problem makes you less vulnerable.

We often overlook the value of communicating better with your teammates. However, all of the arguments over commenting, tabbing, and variable naming all come down to the core need of good communication. It's important to make it easy for other developers to read your code. That could mean adopting company standards or a code framework. In any case, it's part of the professional developer toolkit, and so it goes here.

Just because I didn't mention a specific technique doesn't mean that it doesn't qualify. When in doubt, ask, does it make me more productive, less vulnerable, or more understandable? If the answer is yes, then you're dealing with some sort of professional development tool or technique.

## 2.2 Who Are These Skeptics?

The skeptics are by and large your co-workers who are not using the tool or technique that you want them to adopt. Some don't know about it. Some don't care to know about it. Some know about it and just refuse. By and large, skeptics tend to fall into patterns. I'll go into detail later on the actual resistance patterns.

But what's important to get now about the skeptics is that you need to figure out why they aren't using the technique already or why they rebuff your attempts to introduce it. There are lots of reasons. Some may be technical, some may be political, and some are even personal if you can believe that. The important thing to do is to put yourself in their shoes and try to figure out where they are coming from.

Now, I'm careful to use the term *skeptic* and not something stronger like intransigent, shortsighted, hostile, or even some stronger words you might think of. It's easy in the height of frustration to think of them as adversaries, the opposition, or even enemies. It might feel good to vent about them in that way every once in a while, but don't get stuck there. They are your co-workers and friends, and perhaps you have played this role to someone else. Don't lose sight of that.

## 2.3 Why Do We Need to Sell It?

Selling usually refers to getting people to hand you money for a product. When it comes to selling professional development, though, we're usually but not always looking for another kind of investment. We're usually looking for time or effort. It takes time and effort to learn new things. Sometimes people have trouble seeing the worth of that time and effort, especially if their current tools are inefficiently keeping them working at a frenzied pace. This frenzied pace doesn't give them the ability to step back and see the bigger picture: that they are wasting time using slow methods and tools.

That cost can be a little more subtle sometimes. Programmers tend to define themselves by their language: "I'm a Java developer." or "I'm a .NET developer." Getting a Java developer to try Ruby is more than getting them to spend the time; it's about getting them to rethink their identity, even for a bit. Don't even get me started on trying to get people to try other OS platforms.

It can get even more ephemeral than that. There comes a point when some people think they have gained mastery over their field. Maybe it's when they don't have to look at reference docs anymore, maybe it's after ten years in the field, or maybe it's after an advanced degree. Regardless of the particular milestone, some people think they have all of the answers. You're saying "Something may be an improvement on their methods." They hear "I think you are wrong." If they are wrong now, perhaps they have been wrong for the past few years. Even the most evolved and enlightened people can't always take that, because you are messing with not just their identities but their pride.

In all these cases, you're trying to get more than mere money out of people. You're looking for time, effort, identity shifts, and pride. All of these are more valuable than money. If you think you have to sell to get money, then you'll have to sell even harder to get these.

By now you should have a good introduction of the issues around your tool and technique. You know what it is, you know the people who are in your way, and you know why you need to sell it. However, you need to consider one more important matter: should you be selling your tool or technique in the first place? The next chapter will help us with this issue.

# The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

# Visit Us Online

**Home page for Driving Technical Change**
http://pragprog.com//titles/trevan/
Source code from this book, errata, and other resources. Come give us feedback, too!

**Register for Updates**
http://pragprog.com/updates
Be notified when updates and new books become available.

**Join the Community**
http://pragprog.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

**New and Noteworthy**
http://pragprog.com/news
Check out the latest pragmatic developments, new titles and other offerings.

# Buy the Book

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragprog.com//titles/trevan/.

# Contact Us

| | |
|---|---|
| Online Orders: | www.pragprog.com/catalog |
| Customer Service: | support@pragprog.com |
| Non-English Versions: | translations@pragprog.com |
| Pragmatic Teaching: | academic@pragprog.com |
| Author Proposals: | proposals@pragprog.com |
| Contact us: | 1-800-699-PROG (+1 919 847 3884) |