Extracted from:

# Pragmatic Thinking and Learning

## Refactor Your "Wetware"

# Chapter 1

# Introduction

Welcome!

Thanks for picking up this book. Together, we're going to journey through bits of cognitive science, neuroscience, and learning and behavioral theory. You'll see surprising aspects of how our brains work and see how you can beat the system to improve your own learning and thinking skills.

We're going to begin to refactor your wetware—redesign and rewire your brain—to make you more effective at your job. Whether you're a programmer, manager, "knowledge worker," technogeek, or deep thinker, or if you just happen to have a human brain you'd like to crank up, this book will help.

I'm a programmer, so my examples and rants will be directed at the world of software development. If you're not a programmer, don't worry; programming really has little to do with writing software in arcane, cryptic languages (although we have a curious attachment to that habit).

Programming is all about problem solving. It requires creativity, ingenuity, and invention. Regardless of your profession, you probably also have to solve problems creatively. However, for programmers, combining rich, flexible human thought with the rigid constraints of a digital computer exposes the power and the deepest flaws of both.

Whether you're a programmer or frustrated user, you may have already suspected that software development must be the most difficult endeavor ever envisioned and practiced by humans. Its complexity strains our best abilities daily, and failures can often

be spectacular—and newsworthy. We've smashed spaceships into distant planets, blown up expensive rockets filled with irreplaceable experiments, plagued consumers with automated collection letters for $0.00, and stranded airline travelers on a semiregular basis.

But now the good news (sort of): it's all our fault. We tend to make programming much harder on ourselves than we need. Because of the way the industry has evolved over time, it seems we've lost track of some of the most fundamental, most important skills needed by a software developer.

The good news is that we can fix that right here and right now. This book will help show you how.

The number of bugs programmers introduce into programs has remained constant for the past forty years. Despite advances in programming languages, techniques, project methodologies, and so on, the defect density has remained fairly constant.[1]

Maybe that's because we've been focusing on the wrong things. Despite all these obvious changes in technology, one thing has remained constant: us. Developers. People.

Software isn't designed in an IDE or other tool. It's imagined and created in our heads.

Software is created in your head.

Ideas and concepts are shared and communicated among a team, including the folks who are paying our organization to develop this software. We've spent the time investing in basic technology—in languages, tools, methodologies. That was time well spent, but now it's time to move on.

Now we need to look at the really hard problems of social interaction in and between teams and even at the harder issues of just plain old thinking. No project is an island; software can't be built or perform in isolation.

Frederick Brooks, in his landmark paper "No Silver Bullet–Essence and Accident in Software Engineering" [Bro86], claimed that "the software product is embedded in a cultural matrix of applications,

1. Based on research by Capers Jones via Bob Binder.

users, laws, and machine vehicles.[2] These all change continually, and their changes inexorably force change upon the software product."

Brooks' observation puts us squarely at the center of the maelstrom of society itself. Because of this complex interaction of many interested parties and forces and the constant evolution of change, it seems to me that the two most important modern skills are these:

- Communication skills
- Learning and thinking skills

Some improvement to communication skills is being addressed by our industry. Agile methods (see the sidebar on the following page), in particular, emphasize improved communications between team members and between the ultimate customer and the development team. Mass-media books such as *Presentation Zen: Simple Ideas on Presentation Design and Delivery* [Rey08] are suddenly best-sellers as more and more people realize the importance of simple, effective communication. It's a good start.

But then there's learning and thinking, which is a much harder nut to crack.

Programmers have to learn constantly—not just the stereotypical new technologies but also the problem domain of the application, the whims of the user community, the quirks of their teammates, the shifting sands of the industry, and the evolving characteristics of the project itself as it is built. We have to learn—and relearn—constantly. Then we have to apply this learning to the daily barrage of both old and new problems.

It sounds easy enough in principle perhaps, but learning, critical thinking, creativity, and invention—all those mind-expanding skills—are all up to you. You don't get taught; you have to learn. We tend to look at the teacher/learner relationship the wrong way around: it's not that the teacher *teaches*; it's that the student *learns*. The learning is always up to you.

It's my hope that *Pragmatic Thinking and Learning* can help guide you through accelerated and enhanced learning and more pragmatic thinking.

2. That is, platforms.

**What Are Agile Methods?**

The term *agile methods* was coined at a summit meeting in February 2001 by seventeen leaders in software development, including the founders of various development methodologies such as Extreme Programming, Scrum, Crystal, and, of course, our very own pragmatic programming.

Agile methods differ from traditional plan-based methods in a number of significant ways, most notably in eschewing rigid rules and discarding dusty old schedules in favor of adapting to real-time feedback.

I'll talk about agile methods often throughout the book, because many of the agile ideas and practices fit in well with good cognitive habits.

## 1.1 Again with the "Pragmatic"?

From the original *The Pragmatic Programmer: From Journeyman to Master* [HT00] to our Pragmatic Bookshelf publishing imprint, you will notice that we have a certain preoccupation with the word *pragmatic*. The essence of *pragmatism* is to do what works—for you.

So before we begin, please bear in mind that every individual is different. Although many of the studies that I'll reference have been conducted on large populations, some have not. I'm going to draw on a large variety of material ranging from hard scientific fact proven with functional MRI scans of the brain to conceptual theories, as well as material ranging from old wives' tales to "Hey, Fred tried it, and it worked for him."

In many cases—especially when discussing the brain—the underlying scientific reasons are unknown or unknowable. But do not let that worry you: if it works, then it's pragmatic, and I will offer it here for your consideration. I hope many of these ideas will work for you.

Only dead fish go with the flow.

But some folks are just plain wired differently; you may be one of them. And that's OK; you shouldn't follow any advice blindly. Even mine. Instead, read with an open mind. Try the suggestions, and decide what works for you.

**What Is Wetware?**

wet•ware |ˈwetˌwe(ə)r|: etymology: wet + software

*Noun, humorous.* Human brain cells or thought processes regarded as analogous to, or in contrast with, computer systems.

That is, using the model of a computer as an analogy to human thought processes.

As you grow and adapt, you may need to modify your habits and approaches as well. Nothing in life is ever static; only dead fish go with the flow. So, please take this book as just the beginning.

I'll share the pragmatic ideas and techniques I've found in my journey; the rest is up to you.

## 1.2 Consider the Context

Everything is interconnected: the physical world, social systems, your innermost thoughts, the unrelenting logic of the computer—everything forms one immense, interconnected system of reality. Nothing exists in isolation; everything is part of the system and part of a larger context.

Because of that inconvenient fact of reality, small things can have unexpectedly large effects. That disproportionate effect is the hallmark of nonlinear systems, and in case you hadn't noticed, the real world is decidedly nonlinear.

> *When we try to pick out anything by itself, we find it hitched to everything else in the universe.*
>
> ► John Muir, 1911, *My First Summer in the Sierra*

Throughout this book, you'll find activities or differences that seem to be so subtle or inconsequential that they couldn't *possibly* make a difference. These are activities such as thinking a thought to yourself vs. speaking it out loud or such as writing a sentence on a piece of paper vs. typing it into an editor on the computer. Abstractly, these things should be perfectly equivalent.

But they aren't.

These kinds of activities utilize very different pathways in the brain—pathways that are affected by your very thoughts and how you think them. Your thoughts are not disconnected from the rest of the brain machinery or your body; *it's all connected*. This is just one example (and we'll talk more about the brain later in the book), but it helps illustrate the importance of thinking about interacting systems.

Everything is interconnected.

In his seminal book *The Fifth Discipline: The Art and Practice of the Learning Organization* [Sen90], Peter Senge popularized the term *systems thinking* to describe a different approach of viewing the world. In systems thinking, one tries to envision an object as a connection point of several systems, rather than as a discrete object unto itself.

For instance, you might consider a tree to be a single, discrete object sitting on the visible ground. But in fact, a tree is a connection of at least two major systems: the processing cycle of leaves and air and of roots and earth. It's not static; it's not isolated. And even more interesting, you'll rarely be a simple observer of a system. More likely, you'll be part of it, whether you know it or not.[3]

TIP 1

Always consider the context.

Put a copy of that up on your wall or your desktop, in your conference room, on your whiteboard, or anywhere you think alone or with others. We'll be returning to it.

## 1.3 Everyone Is Talking About This Stuff

As I was mulling over the idea of writing this book, I started to notice that a *lot* of people in different disciplines were talking about the topics in which I was interested. But these were in very different and diverse areas, including the following:

- MBA and executive-level training
- Cognitive science research

3. Suggested by our old buddy Heisenberg and his quantum uncertainty principle, the more general *observer effect* posits that you can't observe a system without altering it.

- Learning theory
- Nursing, health care, aviation, and other professions and industries
- Yoga and meditative practices
- Programming, abstraction, and problem solving
- Artificial intelligence research

When you start to find the same set of ideas—the same common threads—showing up in different guises in these very different areas, that's usually a sign. There must be something fundamental and very important lurking under the covers for these similar ideas to be present in so many different contexts.

There's something fundamental here.

Yoga and meditative techniques seem to be enjoying quite a bit of mainstream popularity these days, and not always for obvious reasons. I noticed an article in an in-flight magazine around October 2005 that trumpeted the headline "Companies Now Offering Yoga and Meditation to Help Fight Rising Health-Care Costs."

Large companies have not historically embraced such warm-and-fuzzy activities. But the meteoric rise of health-care costs has forced them to take *any* course of action that might help. Clearly, they believe the studies showing that practitioners of yoga and meditative techniques enjoy greater overall health than the general population. In this book, we're more interested in the areas related to cognition, but greater overall health is a nice side benefit.

I also noticed that a number of MBA and executive-level courses promote various meditative, creative, and intuitive techniques—stuff that fits in perfectly with the available research but that has not yet been passed down to the employees in the trenches, including us knowledge-worker types.

But not to worry, we'll be covering these topics here for you. No MBA required.

## 1.4 Where We're Going

Every good journey begins with a map, and ours appears in the front portion of this book. Despite the linear flow of a book, these topics are entwined and interrelated, as the map shows.

After all, everything is connected to everything else. But it's somewhat difficult to appreciate that idea with a linear read of a book. You can't always get a sense of what's related when faced with countless "see also" references in the text. By presenting the map graphically, I hope you get the opportunity to see what's related to what a little more clearly.

With that in mind, the following is roughly where we are headed, despite a few side trips, tangents, and excursions on the way.

### Journey from Novice to Expert

In the first part of the book, we'll look at *why* your brain works as it does, beginning with a popular model of expertise.

The Dreyfus model of skill acquisition provides a powerful way of looking at how you move beyond beginner-level performance and begin the journey to mastery of a skill. We'll take a look at the Dreyfus model and in particular look at the keys to becoming an expert: harnessing and applying your own experience, understanding *context*, and harnessing *intuition*.

### This Is Your Brain

The most important tool in software development is, of course, your own brain. We'll take a look at some of the basics of cognitive science and neuroscience as they relate to our interests as software developers, including a model of the brain that looks a lot like a dual-CPU, shared-bus design and how to do your own brain surgery of a sort.

### Get in Your Right Mind

Once we have a better understanding of the brain, we will find ways to exploit underutilized facets of thinking to help encourage better creativity and problem solving, as well as harvest and process experiences more effectively.

We'll also take a look at where *intuition* comes from. Intuition, the hallmark of the expert, turns out to be a tricky beast. You need it, you rely on it, but you also probably fight against using it constantly, without knowing why. You may also be actively suspicious of your own and others' intuition, mistakenly thinking that it's "not scientific."

We'll see how to fix that and give your intuition freer reign.

### Debug Your Mind

Intuition is a fantastic skill, except when it's wrong. There are a large number of "known bugs" in human thinking. You have built-in biases in your cognition, influences from when you're born and from your cohort (those born about the same time as you), your innate personality, and even hardware wiring problems.

These bugs in the system often mislead you by clouding your judgment and steering you toward bad, even disastrous, decisions.

Knowing these common bugs is the first step to mitigating them.

### Learn Deliberately

Now that we've gotten a good look at how the brain works, we'll start taking a more deliberate look at *how* to take advantage of the system, beginning with learning.

Note that I mean *learning* in the broadest sense, covering not only new technologies, programming languages, and the like, but also your learning of the dynamics of the team you're on, the characteristics of the evolving software you're building, and so on. In these times, we have to learn all the time.

But most of us have never been taught how, so we sort of wing it as best we can. I'll show you some specific techniques to help improve your learning ability. We'll look at planning techniques, mind maps, a reading technique known as SQ3R, and the cognitive importance of teaching and writing. Armed with these techniques, you can absorb new information faster and easier, gain more insights, and retain this new knowledge better.

### Gain Experience

Gaining experience is key to your learning and growth—we learn best by doing. However, just "doing" alone is no guarantee of success; you have to learn from the doing for it to count, and it turns out that some common obstacles make this hard.

You can't force experience either; trying too hard can be just as bad (if not worse) than slogging through the same old motions. We'll take a look at what you need to create an efficient learning environment using feedback, fun, and failure; see the dangers of deadlines; and see how to gain experience virtually with mental grooving.

### Manage Focus

Managing your attention and focus is the next critical step in your journey. I'll share with you some tricks, tips, and pointers to help you manage the flood of knowledge, information, and insights that you need to gain experience and learn. We live in information-rich times, and it's easy to get so swamped under the daily demands of our jobs that we have no chance to advance our careers. Let's try to fix that and increase your attention and focus.

We'll take a look at how to optimize your current context, manage those pesky interruptions better, and see why interruptions are such cognitive train wrecks. We'll look at why you need to defocus in order to focus better in the mental marinade and manage your knowledge in a more deliberate manner.

### Beyond Expertise

Finally, we'll take a quick look at why change is harder than it looks, and I'll offer suggestions for what you can do tomorrow morning to get started.

I'll share what I think lies beyond expertise and how to get there.

So, sit back, grab your favorite beverage, and let's take a look at what's under the hood.

## Next Actions

Throughout the book, I'll suggest "next actions" that you can take to help reinforce and make this material real for you. These might include exercises to do, experiments to try, or habits to start. I'll list these using checkboxes so you can check the items you've done, like this:

- ☐ Take a hard look at current problems on your project. Can you spot the different systems involved? Where do they interact? Are these interaction points related to the problems you're seeing?
- ☐ Find three things you've analyzed out of context that caused you problems later.
- ☐ Put up a sign somewhere near your monitor that reads "Consider the context."

**About the Figures**

You may notice that figures in this book don't look like the typical shiny, mechanically perfect drawings you'd expect from Adobe Illustrator or something similar. That's quite deliberate.

From the electronics books by Forrest M. Mims III to the back-of-the-napkin design documents favored by agile developers, hand-drawn figures have certain unique properties, and we'll see why a bit later in the book.

## 1.5 Grateful Acknowledgments

Very special thanks to Ellie Hunt for introducing me to the Dreyfus model and related nursing research, suffering through my disjointed and rambling prose, keeping me on track, and keeping our domestic enterprises running like a well-oiled machine. A regular editor's job is often difficult and thankless, and mere appreciation in a preface really doesn't do it justice. To be editor, mom, and business manager all at once truly takes skill and patience.

Thanks to my friends on the Pragmatic Wetware mailing list and reviewers, including Bert Bates, Don Gray, Ron Green, Shawn Hartstock, Dierk Koenig, Niclas Nilsson, Paul Oakes, Jared Richardson, Linda Rising, Johanna Rothman, Jeremy Sydik, Steph Thompson, Derek Wade, and everyone else who posted their thoughts, experiences, and readings. Their combined experiences are invaluable.

Special thanks to June Kim for his many contributions throughout the book, including pointers to far-flung research and stories of his own experiences, as well as his feedback throughout the stages of birthing this book.

Special thanks also to Dr. Patricia Benner, who introduced the Dreyfus model of skills acquisition to the nursing profession, for her support and permission to quote from her works and for her enthusiasm for learning.

Thanks go to Dr. Betty Edwards, who pioneered the practical applications of lateral specialization in the brain, for her kind support and permission to quote from her works.

Thanks to Sara Lynn Eastler for the index, to Kim Wimpsett for correcting my many typos and often haphazard grammar, and to Steve Peter for implementing a plethora of typesetting tricks.

And finally, thanks to *you* for purchasing this book and beginning the journey with me.

Let's move our profession forward in the right direction, harness our experience and intuition, and create new environments where learning matters.

# 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

**Pragmatic Thinking and Learning's Home Page**
http://pragprog.com/titles/ahptl
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 in the news.

# Buy the Book

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

# Contact Us

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