Extracted from:

The Passionate Programmer

Creating a Remarkable Career in Software Development

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Bookshelf

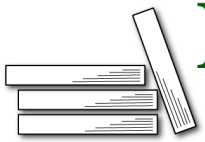
PRAGMATIC LIFE

THE PASSIONATE PROGRAMMER

CREATING A REMARKABLE CAREER IN SOFTWARE DEVELOPMENT

CHAD FOWLER

FOREWORD BY **DAVID HEINEMEIER HANSSON**



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-934356-34-0
Printed on acid-free paper.
Book version: P6.0—June 2012

Tip 51

Avoid Waterfall Career Planning

Back in the beginning of this millennium, an initially small rebellion formed in the software industry. A group of experts in software development started to realize that among them there was a trend forming in the way software projects were both failing and succeeding. In an industry environment in which more software projects were failing than succeeding, they believed that they had discovered a way to do better. The group called themselves the Agile Alliance.

The industry at the time had led itself to believe that the only way to develop software projects was to follow a top-down, heavily planned, rigorous process. Analysts would define requirements in large documents, and architects would create architectures that they would hand down to designers, who would create detailed designs. These designs would be passed to developers who would codify the designs in some sort of programming language. Finally, after months—sometimes years—of effort, the code would be integrated and delivered to a testing group, which would certify it for deployment.

Sometimes some variant of this process would work. If everyone knew every detail of what they needed at the beginning of a project, this kind of planning and rigor could deliver well-thought-out, quality-controlled software. But most of the time, people don't know every detail of what they want out of a big project. The larger and more complex a project is, the less likely it is possible to imagine every feature in detail well enough to create a specification. This kind of process is a *waterfall process*, and that term is almost universally equated with bad process these days.

So, as the members of the Agile Alliance realized, following a heavy process as most organizations were doing back then resulted in well-tested, thoroughly documented software, which was not what the software's users wanted. The rebellion was to create a family of agile methodologies. These were software development processes that were geared toward easy change. Less time was spent up front planning and designing. Software is malleable, so changing it *can* be

cheap. The agile methodologies assumed change as a constant part of software development and adapted themselves to make it as cheap and easy as possible.

It all sounds obvious now. But back then, the adoption of agile processes was a source of conflict and debate. In theory, the idea of detailed planning and rigor sounds obviously right. But in practice, it does not work.

Early in my own conversion to agile methodologies (specifically Extreme Programming), I started to see everything through the lens of agile development. The forces and motivations at play turn out to be more general than just software development. Whenever I had a complex problem to solve, I would realize that an iterative, change-friendly approach to solving it was always less stressful and more effective for me.

Somehow, though, it took me a long time to realize that the most complex project I ever had to manage—the most stressful and most critical—was my career. I had designed my career up front like a software waterfall project. And the same problems that occurred in software projects were starting to happen to me and my career.

I was on track to be a successful corporate vice president or chief information officer. I was doing pretty well on this track. I had rapidly gone from newbie programmer to software architect to manager to director and could easily see myself continuing up the chain. But, successful as I had been, I started feeling like all I was doing was work I didn't like. In fact, the more successful I was, the less likely I was to be in a job I enjoyed.

What I was doing to myself was the same thing heavy processes did to their customers. I was doing an *excellent* job at delivering a career to myself that I *didn't want*.

It was unintuitive to me at first, but the solution to such a problem is to simply *change* your career. That can mean a lot of things. For me, it meant getting back into the deep technology that got me so excited about the information technology industry in the first place. For others I've known, it has meant moving from system administration to software development, moving from an unrelated field into computer programming, or even dropping the profession altogether and doing something else they love.

Just as in software development, the cost of change doesn't have to be high. Sure, it might be hard to go from software testing to being a lawyer. But changing your direction from management to programming, or vice versa, isn't hard. Nor is finding a new company to work for. Or moving to a different city.

And unlike, say, building a skyscraper, changing your career doesn't require throwing away everything you've already done. I spend my days programming in Ruby at the moment, but my experience as a manager or setting up an offshore development operation are constantly relevant and helpful in what I do. My employers and clients understand this and take advantage of it.

The important thing to realize is that change is not only possible in your career but *necessary*. As a software developer, you would never want to pour yourself into developing something your client doesn't want. Agile methodologies help prevent you from doing so. The same is true of your career. Set big goals, but make constant corrections along the way. Learn from the experience, and change the goals as you go. Ultimately, a happy customer is what we all want (especially when, as we plan our careers, we are our own customers)—not a completed requirement.