Best Books of 2008

Adrian Kosmaczewski

2009-01-06

You might remember my beloved mantras: learning a new programming language and reading at least 6 relevant books every year. Following the 2007 edition, here's the list of the 8 books I have enjoyed most in 2008, ordered by a purely subjective and absolutely irrational decreasing preference. I strongly recommend all of them!

Winner: Geekonomics: The Real Cost of Insecure Software by David Rice

Runner-up: The No Asshole Rule: Building a Civilized Workplace and Surviving One That Isn't by Robert I. Sutton, PhD

And 6 more:

- Software Estimation: Demystifying the Black Art by Steve McConnell
- Modern C++ Design: Generic Programming and Design Patterns Applied by Andrei Alexandrescu
- It's Not How Good You Are, It's How Good You Want to Be by Paul Arden
- RESTful Web Services by Leonard Richardson and Sam Ruby
- JavaScript: The Good Parts by Douglas Crockford
- Programming Collective Intelligence: Building Smart Web 2.0 Applications by Toby Segaran

Geekonomics: The Real Cost of Insecure Software

My big winner for 2008. If you don't care about software quality (yet), or if you think that machines aren't already in charge or our world, this book is for you. This has been one of the most hyped releases of 2008, and indeed, it can send a chill down your spine. Geekonomics is a lively catalog of software glitches and disasters, showing users as "crash test dummies", describing a whole industry built upon "end user license agreements", removing liabilities as no other industry has ever been capable of. Even the world of free and open source software is described with strong criticism.

Geekonomics sometimes feels like a desperate plea to apply Deming's Total Quality Management principles in the world of software.

The book is interesting in several fronts. David Rice pledges for the creation of standards of quality, as well as tightening the requirement for certification of practitioners. His views are based on the United States, describing the legal framework of "tort law", the economic foundations of "incentives" for industries, and using comparisons with other economic sectors, particularly with the automotive industry. The book is academic yet entertaining, frightening but instructive, but sometimes falling on the side of sensationalism, in my opinion. I could even say that some of Rice's proposals are downright impossible to achieve, given the particular characteristics of the software industry, and the situation of the legal system in other parts of the world.

In any case, even if I do not particularly agree with all of his views, David Rice is right to emphasize his point strongly, giving concrete proposals, and opening the debate: the book would not have had the same impact if it had been written mildly.

Geekonomics was enlightening: it made me think about the quality of my own work in a completely different way (and prompted me to talk about it at Lausanne's 2008 Barcamp). I think that taking a time of introspection, and reading your own code with different eyes is required to realize that we are, as software developers, responsible for much of what is going on in the world. And this is the major contribution of this book.

The No Asshole Rule: Building a Civilized Workplace and Surviving One That Isn't

Some of my former colleagues will chuckle when they see this entry, because I've flown from a previous job a couple of years ago because of a total, complete, utter, outright and unmitigated asshole. A complete control freak, self-proclaimed genius, irresponsible jackass, who was the reason why 5 people (including me) left the place in less than 3 months.

(Guys, feel free to leave comments below ;)

More seriously, in the software industry it's common to see that great developers are, more often than not, shy or introverted. This fact, coupled with the Swiss' legendary attitude of eternal conciliation and conflict avoidance, has the side effect of creating troubled workplaces all over the country. No wonder this book made the headlines last month in Switzerland. This is a huge problem which is only being revealed right now.

In any case, this book is, together with Peopleware, a gem of teambuilding and a real bag of tricks to avoid turnover and burnouts in your company. An absolute read to everyone involved in the creation of workplaces, in every possible industry.

Software Estimation: Demystifying the Black Art

This book should be a mandatory read for every software developer, in every company, all over the world. It is a true gem, and a book that will become a timeless classic.

How many times have you been asked to estimate a project? To provide a deadline? To approve an estimation done by someone else? This book starts by enumerating the problems related to estimation, including those related to the definition of the word "estimation", the politics and the economics surrounding and defining what is accepted and required, and the common traps where all of us have stumbled upon at least once. Then it provides a catalog of common estimation methods, from the most obvious to the most complex ones, including a description of their relative drawbacks and benefits.

McConnell is the author of Code Complete (which I'm re-reading these days) and Rapid Development (which I plan to read soon). This guy knows what he's talking about, and he provides all the data required to support his claims. I cannot stress this more: if you are a project manager, a developer, a tester, an architect, or deal with software projects in some uncanny way, you have to read this book.

Modern C++ Design: Generic Programming and Design Patterns Applied

I bought this book during the writing of my Master's degree dissertation project. I had heard about it before, but since my project involved a lot of C++ template metaprogramming, I thought that the best idea was to check with the real experts. And boy, I'm happy I did.

This book goes beyond your common programming grounds, whichever your favorite language is. I personally enjoy writing C++ a lot, but that's me, and your mileage may vary. Doing multiple inheritance mixed with template metaprogramming, however, stretches your mind into the realm of what you previously considered esoteric and impossible, and that's precisely the kind of readings that take you a long way forward.

If you are looking for a new way to write your old C++ code, or if you are asking yourself how different are C++ templates from Java or C# generics, read this book. You'll find a lot of interesting stuff in it, with explanations related to an existing library (written by the author) called Loki, which provides the implementation of several patterns explained in the book.

It's Not How Good You Are, It's How Good You Want to Be

This little book (less than 130 pages long) was written by Paul Arden, former creative director of Saatchi & Saatchi, a recognized advertising agency (whose

"early growth was also helped by a policy of settling the invoices from small suppliers as late as possible, while promptly paying large, high-profile companies", dixit their Wikipedia article. No comments.)

Paul Arden provides extremely interesting and pragmatic views on creativity, people management, client relationship management and other stuff; if you happen to work in a web or advertising agency, you should read this book. However, as a software developer, I think that many of his views are still applicable to our own activity, particularly when, like me, you're beginning your own independent path. It is not always obvious how to deal with situations when you're "in charge", and Arden's recommendations do help.

RESTful Web Services

For years I thought that SOAP was the way to go. And then some Rails activists started talking about RESTful this, RESTful that, and well, it tickled my curiosity. It all started with Roy Fielding's doctoral dissertation about network architectures, followed by all the buzz of Web 2.0 APIs, which in the case of most startups took the form of RESTful APIs, found to be much easier to document, use and maintain than their XML-RPC or SOAP counterparts.

I read this book because of a requirement of a project where I worked at the beginning of the year, and this led to several blog posts and even a project, that are still today quite popular. I've been using the RESTful approach for other projects, involving .NET, the iPhone and Cocoa on the Mac, and I would have never thought that doing network-based programming could be this fun. Looking backwards, attempts like SOAP and XML-RPC look clunky, adding layers of unneeded complexity for most projects, and creating a higher barrier of entry for new users of an API.

JavaScript: The Good Parts

Douglas Crockford is head of web development at Yahoo!, and probably the person in the world that knows most about JavaScript. I've been reading his articles since I started working in my Propano project, and then watching his videos as soon as they became available.

His views of JavaScript as the World's Most Misunderstood Programming Language helped me see this (now I think) beautiful language under a new light. And of course, I could not miss reading his short (170 pages) but extremely detailed book. I thoroughly enjoyed it and strongly recommend its reading to anyone dealing with JavaScript in any way.

Programming Collective Intelligence

I love programming books focusing on solutions rather than on specific features of some language; they provide insight on how to solve problems, showing the mathematical background required to do it. This is one of them.

This book can be considered a practical handbook on statistics programming, using Python for the code examples, but providing all the required insight and theory required to understand the logic beneath every code bit. The focus on Web 2.0 applications is clear, and this book has helped more developers than the author might ever know. Social sites are the realm of networks, large numbers, big datasets and complex relationships, and the techniques described in this book can help developers get out more information about their databases. Ever wondered how LinkedIn finds out someone you might know? This book has the answer for you.

What about you?

I would be very pleased if you could leave, in your comments below, the reference to your own preferred books. I am sure I have missed some gems last year, but 2009 is just starting!