H3RALD

Book Review: Making it Big in Software

A handbook for (almost) guaranteed success in IT

by *Fabio Cevasco*

February 2013

Originally published on H3RALD.com

Part I: Fundamentals 2

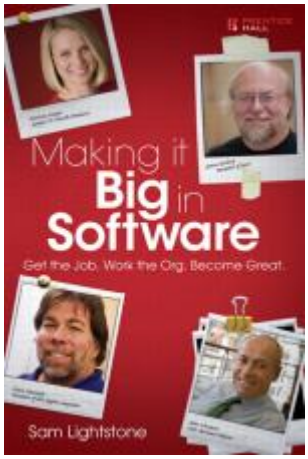
Part II: Leadership 3

Part III: Greatness 4

The Interviews 4

Final Thoughts 4

When this book came out, it was immediately followed by a lot of buzz. Positive reviews started popping up almost instantly, a lot of people blogged about it, it was surrended by a lot of… what's that word again? Oh yes, *hype*. The title pissed me off really: who on Earth wants to title his book Making it Big in Software? Steve Jobs? Bill Gates?
No, just a guy named Sam Lightstone. When I was offered a review copy, I was a bit reluctant to even bother: I thought it was one of those overly-hyped titles that claim to make you famous and successful, but all they do stating the obvious: work hard, be innovative, use your money wisely, etc. Well, this book is not one of them. When I got my copy, I immediately read the author's bio on the second-last page of the book: Sam Lightstone runs a site called Making it Big Careers (again, I got instantly worried by this), *but* also happens to be one of the brightest minds in IBM, a IBM Master Inventor, author and co-author of 30+ patents.



The 17 exclusive interviews with software gurus, visionaries, minor and major deities of the IT world are definitely worth the 24.99$ this book costs *on their own*. This was one of the major selling points of the book itself (as the merry-looking pictures of Marissa Mayer, James Gosling, Steve Wozniak and John Schwarz on the cover suggest), but far from being the only one. The interviews are strategically placed throughout the book, as supporting material for the author's advice: if you don't believe him, you will believe those who *made it*. Anyhow, let's say something about the book itself, shall we?
Making it Big with Software is divided into three parts:

- **Part I: Fundamentals** — all you need to know to get hired. Finish school, learn new things, and get a job in the Software industry.
- **Part II: Leadership** — tips on what to do to start climbing the corporate ladder, from junior to senior manager.
- **Part III: Greatness** — go beyond a successful career and become a luminary in IT, an example for future generations (and earn the big bucks).

Part I: Fundamentals

After two introductory chapters, aimed at answering questions like "Why bother?" or "What do big shots in software do?", the book starts analyzing what graduates get when they get out of school. I was really taken by the following paragraph, outlining the main difference between school and work:

> *[…] although schools encourage students to do their own work, on penalty of expulsion or severe reprimand, professional work is saturated with the ubiquitous mantra of "teamwork." In school, your success depends on individual effort, whereas professional life depends frequently on your ability to work in teams.*

So true. I never thought about it until I read it in this book. And this is a common causes of failure in the workplace: not being able to work in a team. It's understandable: after years of striving to be the best, to do things for yourself, you're suddenly asked to work for and with others.
The author gives junior graduates some useful tips to get a job in software development (or the software industry in general), with some useful tips on how to create a proper résumé, how to survive interviews, the usual. Hell I wish I had this book when I started!
Readers like me who already have a job should not dismiss this part. Maybe skim through the first few chapters, but towards the end there are some useful suggestions on how to build essential interpersonal skills and a nicely-written chapter about *career killers*.

Part II: Leadership

The second part of the book opens with **Chapter 9**, Working the Org, which I found most amusing for the funny, but insightful, *Negotiating 101* section. Again, particular emphasis is put on non-technical skills, which are however essential for success. I particularly enjoyed reading this part of the book, because I could relate to it, being a Technical Leader myself.
Chapter 12 is a must-read, as the author himself says:

> *If you read only one chapter in this book, this should probably be the one.*

If you never read anything about time management, you rhave to read this, as it helps you realize how much time you waste, why, and what you can do to improve the situation. I attended a course on the subject at work, a while ago, and I was shocked to read most of the stuff I learned at that course so tidily organized in no-nonsense prose in this chapter. Granted, it doesn't substitute a time management course or practical experience with managing your priorities, but it is a good starting point.
Chapter 14 deals with *Zen and the critical art of balance [between work and personal life]*. The diagram on page 249 scared the hell out of me. Here it is, transposed in tabular form:

Desired State	Current State
Work: 9 hours	Work: 13 hours
Sleep: 8 hours	Sleep: 6 hours
Travel: 1 hour	Travel: 2 hours
Family & Leisure: 4 hours	Family & Leisure: 1 hours
Chores & Hygiene: 2 hours	Chores & Hygiene: 2 hours

Thirtheen hours? Really? If *you* work 13-hour days then you have to read this chapter *and put it into practice* instantly or you'll regret it. Luckily *I* manage to work most of the time for 8 hours a day (as everyone should, by law).
Another chapter I particularly enjoyed (and will re-read periodically) is **Chapter16**, which contains the best definition of leadership I ever came across:

> *"Leadership is communicating to people their worth and potential so clearly that they come to see it in themselves."*
>
> – Stephen Covey

Again, this chapter teaches you the basics on leadership and management. If you didn't take a course on the subject yet, it's definitely worth a read.

Part III: Greatness

I particularly enjoyed the first two chapters of this last part: **Chapter 17** and **Chapter 18** are about *innovation*, which I found to be the fastest and best way to get noticed in a company.
These two chapters won't teach you to become a genius or an inventor, but they do provide help on the subject: why innovating is important, how to innovate and what to do once your idea gets a shape. The *Patenting*, *Publishing* and *Public Speaking* sections in chapter 8 are useful and practical, and deserve a good read. Again, the book does not go too in-depth, but the author provides just enough information to make you aware of the main issues.
The final chapters of the book felt a bit distant from my current work reality. Business talk, stock options, startups, acquisitions… They may interest some readers with an enterpreneurial mindset, but not me, at least not now. Nonetheless, business and politics pay a very important role in any IT job, so it's wise to be aware of them.

The Interviews

The 17 interviews with software gurus, miracle workers and other extremely successful chaps make up for about the 20% of the book. They are carefully placed by the author in specific places of the book where they make the most sense (well, most of the time). Every person had to answer a similar set of questions, like "How did you get started in software", "How do you stay on top of technology trends and innovation?" or "Technical leaders and executives are famous for being time-strapped. What strategies do you use to stay sane and use your time effectively?".
Every interviews has at least one personal anecdotes. Some feel almost legendary, like the following:

> *In 1967, at the age of 12, I dreamed of making a difference in the field of computer science. I went off to the local IBM office, literally knocked on their door, and said, "I will do anything for the summer-empty trash cans, you name it." They said, "Go away kid." But there was a sales guy who took pity upon me and threw me a nice Fortran IV [IBM Mathematical Formula Translating System] manual, with the expectation that I'd probably read it and get bored and never come back. But much to his surprise, I came back the following Monday and said, "Hey, this is cool! I just wrote a program and I want to run it." The sales guy was so impressed that he found me an open computer to work on where I could teach myself how to keypunch, program, and debug for what I still recall as a delightful summer.*
>
> – Grady Booch, IBM Fellow and Chief Scientist for Software Engineering, IBM Research

Every interview provides at least a good piece of advice for newcomers to the field. The last chapter of the book summarizes the interviews attempting to draw the profile of the successful IT professional: some founded their own companies, other climbed up the corporate ladder, a few contributed with key inventions (email, the Internet, …) that changed society as we know it. Different levels of greatness, and different ways to reach it: this is what this book is really about.

Final Thoughts

Making it Big in Software is very well organized, in its three main parts. Unless you're already the CEO of a multi-million-dollar company, you can learn something from this book, and even if you are, learning how other people *made it* is always beneficial.
It is not a specialized book, and as such it does not go in depth on anything specific. This is a good thing though, because after you read some of the chapters you feel motivated to learn more about this or that

particular topic, skill or problem. In a way, it can be a good surrogate for more specialized books about résumé creation, job interviews, time management, leadership etc.
Overall, I recommend this book to everyone who wants to become successful in the software industry. Success can come to different degrees of course (or not come at all), but if you're motivated enough and interested in your work, it is definitely within your grasp. *Be goal oriented*. It's not enough, but it's a good start.