Making it Big in Software

Get the job. Work the org. Become great.

Sam Lightstone
Program Director & Senior Technical Staff Member, IBM
Founder, MakingItBigCareers.com
e-mail: feedback@MakingItBigCareers.com
Topics

1. What do the big shots do?
2. Getting a job
3. School versus job
4. Tradecraft & innovation
5. Eight ideas for success
6. About the book “Making it Big in Software”
7. Conclusion
Software is the place to be

- Software is a GREAT place to be
  - 2 billion people connected to the Internet
  - 30 billion RFID tags in 2010
  - The world is generating 15 Petabytes of data every data.
  - Virtually every device that consumes direct current electricity also requires software.

- Our world is increasingly interconnected, automated and interactive.
What do “big shots” in software R&D do?

- Define technical strategy
  - Where to spend the R&D money, possibly entire new products

- Innovation
  - New inventions, patents
  - Research

- Represent an organization or company
  - Lecturing at universities, conferences, and across organizations
  - Media interviews: TV, press.

- Lead within the industry
  - Industrial committees
  - Academic papers
  - Conference and workshop chairs, program committees
  - Standards

- Mentor
  - Educate the next generation of leaders
Kuh-ching!!!
Freedom

- Seniority gives you influence over
  - What you work on
  - Where you work
  - (to a lesser degree) when you work

The Jim Gray story

The late computer scientist Jim Gray, was one of the founders of modern database systems and a leader in scalable computing before his mysterious disappearance in a boating accident on a clear, calm day in 2007. Before recruiting Gray, Microsoft had always resisted the pressure to establish R&D sites outside of their core location in Redmond, Washington. Gray just wasn’t interested in living there. No problem. In 1995, they built their new laboratory, called the Microsoft Bay Area Research Center (BARC), around him. Some may believe that Microsoft had decided to finally tap into the incredible pool of software talent and dynamism in the Bay area, but in truth, the move to open BARC was motivated heavily by a desire to attract Gray to Microsoft.

According to Senior Vice President of Microsoft Research Rick Rashid, “If Jim had wanted a lab in Monte Carlo, we would have built a lab in Monte Carlo.”

(Extract from “Making it Big in Software: Get the job. Work the org. Become great.”)
Pleasure and pain

- Gut wrenching painful responsibilities
  - Lay off your colleagues
  - Cut projects
  - Deliberately invest in tactical engineering decisions you know are wrong in the long run

- All jobs have painful responsibilities.

Okay, it’s hell, but it’s hell that matters. In contrast, some of the miserable tasks relegated to junior staff are more likely described as just plain miserable.
School versus job: Life is a fishbowl

- **School**
  - Do your own work or you are expelled
    - Announce that you shared, and you accelerate your expulsion
  - Fixed workload given to you
  - Constant feedback (assignments, mid-terms, exams)
  - Master what is known

- **Corporate work is the opposite of school!**
  - Not about “fair”
  - Collaborate or you are fired
    - Hide that you shared, and you damage yourself and others
  - Unstructured (unlimited) workload
  - Infrequent feedback (possible once a year?)
  - Create what doesn’t yet exist
Getting a job. It helps to understand the process

- Companies collect resumes. Thousands.
  - Human Resource groups and hiring managers must attempt to filter these.

   Your mission? In a pile of a trillion resumes… make yours stand out.
It helps to understand how the process works

- Most new grads have pretty similar resumes. Sigh.
  - Highlight the difference not the similarities.
  - Awards, distinctions, are CRITICAL.
  - Go as far back as early high school if you need to.
  - Show passion not just activity.
    - Do you program for fun? Programming and math contests?

- Grades
  - Grades are a poor indicator of job performance, but they may be the only differentiator a recruiter has for selecting people for an interview.

- School prestige
  - Matters. Companies hire from specific schools based on reputation.
  - Rightly or wrongly MIT has a better reputation than Lakehead.

- Use connections.
  - Connections shouldn’t affect whether you are offered a job, but they can dramatically affect whether you are considered.
  - Unfortunately in some companies, connections do affect whether you get a position. Nepotism.
Education

- **Bachelors**
  - Needed for many jobs

- **Masters**
  - Won’t make you a better programmer
  - Improves writing skills dramatically
  - Thesis option will give you key research skills

- **PhD**
  - Won’t make you a better programmer (some evidence it hurts)
  - Key research skills
  - Critical for some jobs
    - Research
    - Teaching

- **Grad school impact on salary**
  - Initial minor increase - largely compensates for lost time
  - Intangible benefits are more important to consider
# Who would you hire?

<table>
<thead>
<tr>
<th>Bob Smith</th>
<th>John Doe</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E-mail:</strong></td>
<td>E-mail: <a href="mailto:JohnD@rogerst.com">JohnD@rogerst.com</a></td>
</tr>
<tr>
<td><strong>Education:</strong></td>
<td><strong>Education:</strong></td>
</tr>
<tr>
<td><strong>Employment History</strong></td>
<td><strong>Employment History</strong></td>
</tr>
<tr>
<td>2005-2008 Senior Architect, Microsoft</td>
<td>2005-2008 Senior Architect, Microsoft</td>
</tr>
<tr>
<td>2002-2005 Senior Programmer, Microsoft</td>
<td>2002-2005 Senior Programmer, Microsoft</td>
</tr>
<tr>
<td><strong>Awards:</strong> ACM programming award 1995</td>
<td><strong>Awards:</strong> ACM programming award 1995</td>
</tr>
</tbody>
</table>

Distinctions:
- Coinventor of 12 software patents
- Coauthor of 15 scientific papers
- Founder, IEEE committee on Green Computing

Interests & hobbies
- Sky diving, fencing, triathlon, Celtic harp


Awards:
- ACM programming award 1995
Why student positions dramatically improve your odds

- For the employer recruiting is risky
  - How much can they tell about you from an interview?
  - Academic results correlate only loosely with professional potential.
  - Teamwork, work ethic, productivity, all very hard to assess in an interview.

- Known quantity is better than a potential
  - “Co-op and Intern positions are NOT just a foot in the door. They’re two legs, and a shoulder into the corporate suite.”
Who is more than more than what
(from Making it Big in Software: Get the job. Work the org. Become great.)

Who you work with:

“Try to work with the smartest people you can find, since they challenge you to think and work better…. When you work with really top-notch, smart engineers, it will fundamentally change the way you think and the way you program.”

-Marissa Mayer, VP, Search Products, Google Inc.

Who you work for:

“…always make sure you’re in an environment that leaves you feeling empowered to fully use your talents, and that you feel appreciated for the work you do.”

- Diane Greene, Co-founder and past CEO of VMWare
Skills

...makin’ it big
Know your tradecraft

- Know methodology
  - Skills
  - Theory

- Tools and languages
  - How it’s done

- Follow the research
  - What’s emerging

- QA skills
  - Basics: review methodology, unit test, FVT, SVT, stress, PQA
  - Statistical testing
  - Software Reliability Engineering Process
  - Formal methods
  - Code & branch coverage theory
  - Quality Measurement, Control, and Management
  - Risk analysis, etc.
Programming language popularity

(Data from Making it Big in Software: Get the job. Work the org. Become great.)

[Diagram showing the popularity of various programming languages, with Java at 18.8% and C++ at 13.4% being the most popular, followed by languages like Python, C#, and SQL at around 6% to 7% each.]
Try to differentiate yourself.

Try to differentiate yourself. For example, when Java exploded in the mid-1990s, everybody became a Java programmer, and the market became flooded with cookie-cutter Java programmers. It’s really hard for people to stand out as something that isn’t easily replaceable in that world. My whole career I have tried to stay away from that. Operating system internals, while not considered particularly sexy or part of the mainstream, have allowed me to stand out because of the relatively few people who go into that and the perception “Wow, that’s really hard.” Stay away from the mainstream and the crowds, and find something that is gonna be stable—not just flash-in-the-plan technology.

Mark Russinovich,
Microsoft Technical Fellow
Windows Architect
Recommended for every programmer

1. Learn at least 4 different programming languages and at least 4 different data formats (such as JPEG, XML, delimited text, and MPEG).
2. Develop software that is suitable for at least a thousand people to use concurrently.
3. Develop software that can scale to more than 1TB of data.
4. Work on a project with more than ten programmers.
5. Work on extending code that someone who is no longer available to ask questions of wrote more than five years ago.
6. Fix at least 40 defects in code you did not author.
7. Write code that supports international languages, including UNICODE input, and more than one language of generated user output (error messages, GUI text, and so on).
8. Study the performance characteristics of the following:
   - Data fetched from memory with and without a CPU cache miss
   - Reads of consecutive blocks from disk versus random I/O seeks
   - Large block I/O versus small-sized I/O
   - Three popular languages (such as Java, C/C++, and PHP)
Innovate

- Seek out hard problems
- Keep up with the research
  - IEEE
  - ACM
  - DBLP
  - Know the major conferences in your field, and follow them.
  - Follow research in your field at major universities.
- Don’t be afraid to publish and patent.
- Form personal relationships with the experts.
  - Publishing helps.
Do executives care?

- In 2006 the IBM Institute of Business Value surveyed 968 CEO’s with the goal of understanding CEO priorities. The study showed that CEOs felt growth through innovation that matters, is key.

- In 2005 McInsey Quarterly survey of 9,354 global executives.

![Diagram showing CEO priorities]

*Source: March 2005 McKinsey Quarterly survey of 9,345 global executives*
Innovation is correlated with job success

- Some facts as of Nov. 20 2007:
  - IBM DB2 senior technical staff
    - Senior Technical Staff Members and Distinguished Engineers
  - 13 STSMs and DEs in the list
  - 100% have filed patents
  - 62% are Master Inventors
    - (12 patent filings or more)
Eight important ideas
Idea 1: Dress for success. *wear running shoes*

Is this a spontaneous photo? Observe who is wearing what – and why.

Making it Big in Software. Get the job. Work the org. Become great.
Idea 2: Time management

- Two guiding issues:
  - What do you want to accomplish (goal centric)
  - How will you use your time to accomplish it (time centric).

- Both goal centric and time centric are critical and complimentary for managing your time well.
Idea 2: Goal centric time management

Imagine you are dropped into this scene with a canoe and a paddle. A disembodied voice says to you... “Go”.
Idea 3: The Law of Promotability Inversion

- Contribution
  - Yes, but hard to assess

- Bonus points
  - Yes, quantifiable
    - Patents
    - Publications
    - Public speaking
    - Customer advocacy
  - Critical for leadership and executive roles

- Expertise
  - Yes, easier to assess than contribution
  - Everyone knows who the experts are

In theory the only thing that matters

In theory should matter only slightly

Should barely matter at all in principle
Idea 4: The Value of Measuring Value

2008 measurements of DB2 compatibility for the PL/SQL language

98% compatibility

Although we built the tooling to help our customers measure compatibility, we found several side benefits:

1. Quantify your success inside your own company
2. Demystify your product planning.
3. Improve marketing.
4. Raise team morale.
Idea 5: Why startups fail
(There are many reasons.... this is just one worth reflecting on)

- Good business needs more than a great idea.
- Smart startup look for talented business leaders to join them
- Yes, there are exceptions
  - Mark Zuckerberg
  - Bill Gates
  - Steve Jobs
Idea 6: 99% perspiration

“My personal pet peeve is how many people think the hard part is in the “big and hard problems” or in some fluffy but important-sounding thing like “innovation.” In fact, all the real work is in getting the details right. It’s that “1% inspiration, 99% perspiration” thing. People seem to think that inspiration is the much bigger and important part of the two, but I’ve come to believe that while it’s important to have inspiration, where people actually stumble is when they can’t execute on that inspiration. Inspiration isn’t that rare in the end, but people who have it and then actually follow through…that’s rare.”

-- Linus Torvalds, Creator of the Linux kernel
Idea 7: The centrality of fun

- You will always perform your best, be your most productive, and impress others most when you are working on something you enjoy.

- A successful career is a balance between what you really want to do (e.g. be a Rock Star) and what is practical (e.g. be an Accountant).

  Compromise: computer science?
The gurus all agree
(quotes from “Making it Big in Software: Get the job. Work the org. Become great.”)

“If you are entering the field, find something that you really love doing and get excited about doing, so that you almost feel as if you should be paying them to come to work. I think you need to be happy to be successful, not successful to be happy.”
—David Vaskevitch, Microsoft CTO

“Be sure that you like what you do at work and that you like the people you work with—you’ll have to live with them for a long time.”
—Bjarne Stroustrup, inventor of C++

“The more you can merge what you want to do with your job, the better.”
—Steve Wozniak, inventor of the Apple computer

“I believe that I just work better if I enjoy what I’m doing. I suspect that if anybody wants to be ‘the best’ at whatever they do, they have to realize that it takes decades of hard work. And the main way to actually keep doing decades of hard work is to simply enjoy it so much that you don’t want to stop.”
—Linus Torvalds, original author of GNU/Linux

“…follow your passion. Most of all, be sure you have fun in the process.”
—Grady Booch, IBM Fellow

“…if you can find a job that you really like, the better you’re gonna be. You’re gonna be more productive. You’re going to be happier. You’re going to be more satisfied in general with what’s going on.”
—Ray Tomlinson, inventor of email

“Follow your bliss.”
—Jon Bentley, author of Programming Pearls

“For me, work is really enjoyable, making it hard to define where work ends and fun begins.”
—Marissa Mayer, Google VP

“That’s what’s great—my career is a lot of fun. I don’t view it as work. Computers have always been my hobby. The fact that I get to go to work and work on my hobby and get paid for it is just fantastic.”
—Mark Russinovich, Microsoft Technical Fellow

“What is really important is to make sure that they are growing, contributing, and enjoying what they do.”
—John Schwarz, CEO Business Objects

“My number one piece of advice is to have fun.”
—James Gosling, inventor of Java, Sun VP and Fellow
Idea 8: Making it Big is subjective

- It’s important to have your own idea of success and make sure you know what you’re going after.
  - You can use someone else’s model, but if you get there and are unhappy with where you are, or who you are, what’s it worth?

- What will really satisfy you professionally?
  - Challenge, people, money, impact, freedom…

- The ancient Greeks understood it. Inscribed in the courtyard of the Temple of Apollo at Delphi was the simple aphorism:

  KNOW THYSELF
About the book

Making it Big in Software:
Get the job. Work the org. Become great.

Part I Fundamentals
• Chapter 1 Making It Big
• Chapter 2 What Good Software Is Really About
• Chapter 3 School Versus Job
• Chapter 4 Mission Impossible? Getting a Job in Software
• Chapter 5 Making the Most of the Early Years As a Software Developer
• Chapter 6 Essential Skills. Some Are Even Technical
• Chapter 7 The Sweet Science of Software R&D
• Chapter 8 Career Killers

Part II Leadership
• Chapter 9 Working the Org
• Chapter 10 Successful Software Project Proposals
• Chapter 11 Career Advancement
• Chapter 12 Time Management
• Chapter 13 Avoiding Software Development Overruns
• Chapter 14 Zen and the Critical Art of Balance
• Chapter 15 Secret Insights on Software Project Management
• Chapter 16 The Big Leagues: From Medium-Shot to Big-Shot

Part III Greatness
• Chapter 17 Leadership in Software Innovation
• Chapter 18 The Big Leagues: From Big-Shot to Visionary
• Chapter 19 If I Knew Then What I Know Now
• Chapter 20 Going Out on Your Own: The “Software Startup”
• Chapter 21 Compensation: Kuh-ching!
• Chapter 22 Making It Big?
Making it Big in Software
Get the job. Work the org. Become great.

17 exclusive interviews

Exclusive interviews with

Steve Wozniak
Inventor, Apple computer

John Schwarz
CEO, Business Objects

James Gosling
Inventor, Java programming language

Marissa Mayer
Google VP, Search Products and User Experience

Jon Bentley
Author, Programming Pearls

Marc Benioff
CEO and founder, Salesforce.com

Grady Booch
IBM Fellow and co-founder, Rational Software

Bjarne Stroustrup
Inventor, C++ programming language

David Vaskevitch
Microsoft CTO

Linus Torvalds
Creator, Linux operating system kernel

Richard Stallman
Founder, Free software movement

Peter Norvig
Google’s Director of Research

Mark Russinovich
Microsoft Fellow and Windows Architect

Tom Malloy
Adobe Chief Software Architect

Diane Greene
Co-founder and past CEO of VMware

Robert Kahn
Co-inventor, the Internet

Ray Tomlinson
Inventor, email

Sam Lightstone
Praise for Making it Big in Software

“For me, reading this book was a blissful experience from cover to cover.”
Mike Riley, Dr. Dobb’s

“What makes the book particularly attractive was Sam’s willingness to go out and interview other software stars, including Google’s Marissa Mayer, Java inventor James Gosling and Apple founder Steve Wozniak. The interviews alone are worth the price of admission.”
Eric Lundquist, eWeek.com

“‘Software is an amazing place to build a career.’ If you agree with this sentiment that opens Sam Lightstone’s book you are likely to enjoy reading it and find it motivating.”
Sue Gee, i-programmer.com

“I enjoyed reading the book and have a whole list of discussion points from it. I highly recommend you buy your own copy!”
Jeanne Boyarsky, JavaRanch

“… I'm impressed by how empowered I am after reading just the first few chapters! Seriously, I feel like coding the equivalent of punching a bear in the face!”
Chris Toohey, Dominoguru.com
Useful links on Making it Big in Software

For those who are interested in the topic of software careers and career success, here are some useful links related to the book:

Book reviews:

Groups and blogs:
- Join the Facebook Fan Club for 'Making it Big in Software'
- Join the LinkedIn group for 'Making it Big in Software'
- Subscribe to the Making it Big Careers blog

Details on Amazon.com:
- See book details and reviews on Amazon.com  [http://amzn.to/b08auR](http://amzn.to/b08auR)
Conclusions

- Building a successful career is an art form
  - Technical skills count. Soft skills count
  - The mix is your choice

- Understand the way businesses and organizations work

- Set your own course. Don’t let the river carry you.

“Be goal-oriented... You don’t need to have had courses... You can do it if you’re smart. You can pick up some books, start working on some papers, and figure out how you can solve a problem. Whatever it is, just trust yourself that you don’t need a book— you can write the book.

...Keep in mind that almost all the revolutionary great advances, the big home runs in technology, come from very young people in school or not so many years out of school. You’ve got the energy to follow it through...”

-- Steve Wozniak, Inventor of the Apple computer, cofounder of Apple Inc.
(from Making it Big in Software: Get the job. Work the org. Become great.)