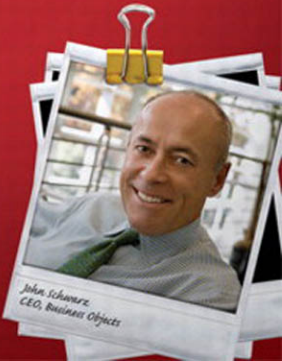
PRENTICE HALL

Making it Big in Software

Get the Job. Work the Org. Become Great.



Sam Lightstone

Photo of Steve Wozniak by Al Luckow.
Photo of Diane Greene by Sunny Scott.

Library of Congress Cataloging-in-Publication Data

Lightstone, Sam.
Making it big in software : get the job—work the org.—become great / Sam Lightstone.
p. cm.
Includes index.
ISBN 978-0-13-705967-6
1. Computer programming—Vocational guidance. 2. Computer software industry—Vocational guidance. 3. Computer software developers—Interviews. I. Title.
QA76.25.L54 2010
005.1023—dc22

2010000235

ISBN-13: 978-0-137-05967-6
ISBN-10: 0-137-05967-1

Text printed in the United States on recycled paper at R.R. Donnelley & Sons, Crawfordsville, Indiana

First printing March 2010

Associate Publisher
Greg Wiegand

Senior Acquisitions Editor
Katherine Bull

Development Editor
Kendell Lumsden

Marketing Manager
Judi Morrison

Publicist
Heather Fox

Technical Reviewers
Joseph Hellerstein
Rasekh Rifaat
Danny Sadinoff

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Hansing Editorial Services

Indexer
Publishing Words

Proofreader
Language Logistics, LLC

Publishing Coordinator
Cindy Teeters

Cover Designer
Alan Clements

Compositor
Nonie Ratcliff

Manufacturing Buyer
Dan Uhrig

Preface

"Thus the sage Chaninah would say:
Much have I learned from my teachers and even more from my friends but from my students I have learned the most of all"
—Ethics of the Fathers, circa 200 CE

You went to university to study a profession, but they were hell-bent on giving you an education instead. Unfortunately, there's a gap between the formal education we receive in school and the skills we need to build truly successful careers. Not only are many of the basic skills for professional growth not taught in school, but to a large degree they aren't taught anywhere. The dynamic and somewhat bohemian quality of the software industry introduces unique career challenges. Our industry includes many of the trappings of corporate American culture, but many odd divergences. It's a domain where teenage hackers compete head-to-head with MIT Ph.D. graduates and where crinkled T-shirts and unlaced running shoes coexist with stock options and executive titles. *Making it Big in Software* is my attempt to share strategies for traversing these unique dynamics and maximizing your professional potential. This book also includes interviews with some of the most influential software innovators and leaders of the past 30 years—people who literally changed the world.

In 1991, I was a fourth-year electrical engineering student at Queen's University, trying to finish up my degree and get a job. Every Friday afternoon the electrical engineering school invited a guest speaker to inspire and enlighten our young, impressionable minds. These talks covered a wide range of technological topics; we discussed everything from high voltage transmission lines to CMOS VLSI circuit design. I confess that most of these sessions were less than inspirational for me. One day, a guest speaker arrived with a radically different message. He spoke to us about "life in the real world" and what we could expect after we graduated. I was riveted. This is the only fourth-year seminar from which I still have the notes. Most of my classmates, like me, were in the dark about what our lives might be like after graduation. Even those students who had managed to find summer positions in

engineering were limited in their experiences by the normal constraints of student positions.

I decided then that if I ever could, I would return the professional courtesy and volunteer to speak to university students. In the late 1990s and early 2000s, I began a series of career talks at leading universities. The lecture notes from my talks, often delivered to capacity-filled auditoriums, became the basis for this book. Some of the ideas are my own, but many are culled from the leading business thinkers and software development evangelists of the past three decades. Hopefully, this book succeeds in providing you what school and daily work life generally cannot: the tools with which to make it BIG.

Audience

This is a book for software professionals of all ages and levels, whether they're just trying to break into the field or have decades of experience. Writing a book with that range isn't easy; every age group and community has a different set of concerns that affect what's interesting to them. After nearly two decades of recruiting, managing, and mentoring software professionals, there are some common themes I've found that cross most boundaries of age and experience. I've tried to make those themes central to the book. I hope this book will also be of interest to students and teachers of computer science, providing a colorful look into the less mathematical aspects of the profession.

Organization of This Book

The book is divided into three major sections: fundamentals, leadership, and greatness. "Part I, Fundamentals," discusses the major building blocks of a great career in software, including the fundamentals of good software products, major skills and leading programming languages, how to land a job, and how to operate effectively within a software team (including some pitfalls to avoid). "Part II, Leadership," takes the reader into a series of topics around driving and leading change and operating in a much more harried professional environment. Leaders, almost by definition, need to multitask and drive parallel agendas forward while deflecting myriad forces of negativity. This section covers important topics related to "working an organization," how to build and pitch successful project proposals, career advancement, time

management, avoiding software project overruns, work-life balance, and higher-level management and leadership insights. "Part III, Greatness," discusses topics around innovation—how to reach the pinnacle of the profession (becoming a software visionary or guru), how to start your own business, a review of compensation rates, and a retrospective on things I wish I had known earlier in my career.

Between the chapters you'll find a number of interviews I did with leading personalities. These interviews deliberately include a mix of business executives, researchers, and industry leaders. I feel it's important for readers to get perspectives broader than my own. To enhance the structure of the book, the interviews are all marked with a gray bar along the leaf edge of their pages, including the interviewee's name. I've made a moderate effort to place the interviews near related chapters (for example, two startup sensations, Marc Benioff CEO and founder of Salesforce.com and Diane Greene past CEO and co-founder of VMware, frame the chapter on software startups). Some interviews didn't have a clear correlation to a specific topic, so there's an admittedly arbitrary aspect in the placement of a few. It's absolutely not the case that interviews placed in the Fundamentals section are any less profound than those in the Greatness section. I think it's fair to say that everyone I interviewed has reached the heights of the profession. After all, that's what made them such interesting people to speak with.

Feedback

If you have feedback on the book or find any errors, please contact me. You can send feedback to feedback@makingitbigcareers.com. Although I can't guarantee a reply to every email, I'll make sure that every email is read.

Sam Lightstone

To learn more about *Making it Big in Software*,
including upcoming events and additional publications,
please go to:
www.MakingItBigCareers.com

CHAPTER 3

School Versus Job

"Every person takes the limits of their own field of vision for the limits of the world."
—Arthur Schopenhauer (1788–1860)

School is very different from professional work. Some people stumble in their early professional careers by failing to transition from the school environment they've been immersed in for nearly twenty years into the brave new world of software professionalism. Students operate in a world of highly constrained, well-defined workloads (though it certainly doesn't feel that way when you're a student). As a student your scope is limited to a set of courses that are intended to address your professional needs upon graduation. In short, school is about learning. Professional life is more about getting things done, being both productive and innovative in a way that jells with the people you work with. It's not uncommon for junior staff to treat the workplace as though it were simply the next phase of school. Few things could be more disastrous.

I also believe that a great weakness of many middle managers is failing to understand the differences between the academic and corporate skills requirements, leading them to hire the best and brightest students rather than the best and brightest professionals. A good student is usually smart, hard-working, and conscientious. But success in scholastics doesn't guarantee an ability to innovate, learn independently, work in a team, or show leadership—which are all critical skills for professionals. Hiring the wrong skill set means hiring less effective people, and when that phenomenon is multiplied over many people in an organization, the net result is a marked reduction in the effectiveness and talent of the entire organization. Put another way, hiring the wrong people for the job is a bad way to run a company. On the flip side,

when new graduates position themselves for the workplace, assimilating the culture and needs of a professional life, and managers learn to recruit the best employees for their needs rather than the best students, the mixture forms a potent brew for the rapid success of individuals and organizations alike.

Limited Field of Vision

Whether you're in school studying the art and science of computer programming or working as a software professional, you exist and function in a largely constrained and somewhat artificial environment. Life is a kind of fishbowl. Fish in a bowl can swim and explore, up to a point, but their view of the world is remarkably limited. The fish sees almost nothing beyond the glass of the fishbowl, in part because its eyes can't focus that far and in part because of the diffraction of light as it crosses the boundary from water to glass to air. If the fish could see beyond the fishbowl, at most, its view of the world would be limited to the room the bowl resides in: a few chairs, a sofa, a bookshelf. The real world outside of the fishbowl is far larger and more profound than the tiny colored glass stones and rocks that have been artificially placed in the fish's manufactured domain. But, for better or worse, the fish is blissfully unaware. Software professionals are fish, too.

School Is a Fishbowl

School is a highly artificial environment where the workload is relatively well controlled, where all participants have similar work challenges, and where people are encouraged to do individual work. If you are a student, you probably think I've completely forgotten what school is all about because what I've described is nothing like the world you're living in. Very true, but it's a relative comment. Even though some professors demand a far heavier workload than others, the reality is that these variations are small compared to the variation that would occur without a fixed curriculum, such as in the environment outside of school. You'll protest: "But if there were no set curriculum, that wouldn't be fair!" and you'll be 100% correct. That's one reason schools must have a set curriculum—because school needs to be fair. The process of making school fair leads to a wide array of artificial constraints and behaviors that, although fair, necessarily create a framework of a highly constrained fishbowl.

Second, 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.

Finally, very little that students experience in school is directly influenced by market pressures. There is some indirect influence, as the market drives new technologies to the fore, which, over time, influences curricula. Professors get involved in new technologies and introduce these themes into the courses they teach. Over time, the system of professors, industrial grants, the drive for publications, and review committees will ensure that the educational process stays reasonably connected to current industrial trends. All the while, the student body is happily oblivious.

Welcome to the fishbowl. While you're in it, you won't be able to see too far past the glass. That's fine, as long as you realize that there is a world outside the fishbowl that's a rather different place, then you're well positioned to try to learn more about it.

Industry Is a Fishbowl

How in the world could industry be considered a fishbowl? Isn't industrial work the definition of the real world? It's time for a reality check! The software industry is sensationally complex, and it requires a broad range of skills and disciplines to construct a successful business (the larger the enterprise, the more this is true). People become specialized and focus on a narrow scope of the industrial machine. Every employee has a domain. High up in the organizational hierarchy, people working on the big picture can't possibly know all the little details. At the other end, the detail workers, down in the trenches, have a very hard time understanding the broader scope of every project that is related to them.

Second, if you're like most software developers, your compensation is only loosely tied to the market success of the projects you work on, unless you are an owner in the company (such as owning stock or stock options) or have a compensation plan that is directly tied to quotas or business performance. Similarly, your compensation is only loosely connected to years of service or the technical depth of your job.

Your compensation at work will never directly increase with your productivity or the success of your product. Employees who work five times more will not get paid five times more than others. If your company revenue soars and revenues increase by tenfold, don't expect employee salaries to increase by a factor of ten. If this sounds discouraging, it shouldn't be—remember, the sword cuts both ways. Just as you are unlikely to see your salary grow 300 times in size when your product begins to sell and earn 300 times more revenue, your salary will not be cut to 1/300th if the inverse befalls you. And

although the most productive and valuable people might not get compensated proportional to their contributions in a literal measurement, they do get paid substantially more. Over time, they will accrue much more accomplishment, fame, money, and freedom of movement for their efforts.

Professional software positions almost always limit the software developer from the sales and marketing world, and in larger companies, software developers can be heavily isolated from customers, business strategy, and product planning. In short, it's a controlled and not truly reflective environment. So it's a fishbowl—what you see is not a reflection of reality.

By understanding and being sensitive to the artificial limitations of your environment, you can begin an active effort to extend your skills and expand your professional potential.

Leveraging the Differences

These fishbowls are very different both in style and in kind. The most dramatic difference is the approach to collaboration and teamwork. School teaches you technical skills in software development and software engineering and some teamwork and social skills along the way. The 18 or more years you spent in school deeply entrench within you expectations and values that are sometimes at odds with the expectations of the workplace. For example, at school we are taught that our work must be our own and that reusing the work of others is a serious crime that can lead to suspension or expulsion from the institution. Conversely, in a professional software development company, few things propel a software development project more effectively and reduce time to market than code reuse. Code reuse is an objective in the workforce but a serious liability at school. Similarly, at school, people are required to do their own work, except in the cases of a few group projects. Within corporate software projects, people are expected to work together and help others with some fraction of their time even when they are not directly assigned to a project. The scope of teamwork ranges from small groups to team efforts measured in dozens or even hundreds of people working toward a collective goal to deliver a project or a product.

Many new software developers cripple their careers by clinging to the entrenched ideology they were immersed in for years that "your work must be your own." For good reason: From early childhood until our mid-20s, this is the message we are all consistently given and ordered to follow. Post-graduation, this edict dissolves and is replaced by the belief that work should *de rigueur* be shared and collaborative. It's critical to your success that you rapidly adopt a collective model for success, code and documentation reuse, and

skills sharing. You should collaborate, share, and team with others as much as possible and give as many people credit for that effort as is reasonable.

Table 3.1 lists some other key differences in the school versus work paradigm.

TABLE 3.1 Differences Between School and Industry

CHARACTERISTIC	DIFFERENCE
Drivers and shapers	School requirements are defined by a committee of academics who define the course curriculum. Successful businesses are always driven by market demands: customers and business climate.
Freedom of motion	Within a computer science (or related) major, the range of courses is fairly limited. You can usually take a few electives each term, but they represent a small diversion from your major. In contrast, the range of what you can work on in industry is profound, crossing deeply technical development work, research, business development, management, marketing, and sales. Even within the technical space, there is a huge range of roles, from quality assurance to development, from architecture to innovation and new product generation.
Innovation	The level playing field that school provides limits the opportunities for technical innovation. Exceptions exist, particularly in graduate school thesis programs, which allow students an opportunity to do independent research. By contrast, industry thrives on innovation; it is the distinguishing feature that separates one high-tech company from another.
Interpersonal networking	Networking is useful at school, especially if you need help from classmates to stay on top of assignments. The best and brightest students don't rely heavily on networking with others. In industry, social networking is essential because it connects you to people and ideas that are necessary to help channel the organizational energy of the business.
Leadership	School is largely a level playing field, and the institution works hard to create a common circumstance for all the students within a specific major. Although some opportunities for leadership exist within school (captain of a sports team or student council), schools generally are hierarchy free within the student body. Students do not lead other students. Within industry, leadership skills and leadership roles are prized and encouraged.
Learning and doing	School is about learning, and work is about producing. It's true that school requires students to produce and industry requires employees to learn, but the focus is heavily inverted.
Payment	You pay to go to school, but you get paid to work. That means your college or university works for you, but after graduation, you work for your employer.

TABLE 3.1 Differences between School and Industry (continued)

CHARACTERISTIC	DIFFERENCE
Productivity	Educational institutions reward students with grades. The reward for doing well is a higher grade on a project or an exam. Your GPA is the measure of your success. In industry, the measurement is "contribution," which loosely translates to "productivity" in the broad sense of producing code, designs, technical innovation, new business, improved organizational efficiency or capacity, and so on. Businesses value productivity, whereas educational institutions do not.
Recognition	In school, your evaluation is designed to be fairly quantitative and measurable. Your grades on assignments, tests, and exams define your GPA. Within a professional software development organization, people have widely varying roles, and the evaluations become much more qualitative and subjective.
Risk and impact to others	The risks of messing up your education are serious, but they predominantly affect you and your family. In industry, large projects depend on the skill of the engineering team, and if you screw up, the consequences can affect the entire corporation, many of its employees, and a broad range of customers and business partners.
Personal portfolio	Neither school nor industry requires you to have a personal portfolio of any kind. By "portfolio," I mean a collection of externally visible accomplishments. Having an impressive portfolio won't help you be a better student (though, in some cases, it can help buttress your candidacy for scholarships or graduate school admittance). In a professional setting, however, having a portfolio of major innovations, successful programming projects you've delivered on time and with high quality, trade publications, scientific papers, public speaking engagements, and positions on industrial committees can help propel your career and open new opportunities for collaboration and engagement.

The greatest challenge of your first few years in industry is to unlearn what the educational environment of the past 18 years or more has drilled into you. Understand that you have entered a brave new world with radically different rules of engagement.

Unquestionably, school and industry both require a lot of hard work and are very competitive environments. Many career paths in fields other than software have a slower pace of learning and change following graduation. The rules of the game change less frequently in most professions. The software industry is distinguished by its dynamism and rapid evolution. Everything can change in a few years. The rapid changes that characterize high tech mean that software professionals are necessarily life-long learners. One thing is certain: If you were hoping for a gently loping career that largely takes care of itself, something akin to a warm, comfortable bath, then software is the wrong place for you. Expect a wild ride.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q–R

S

U–V

W–Z