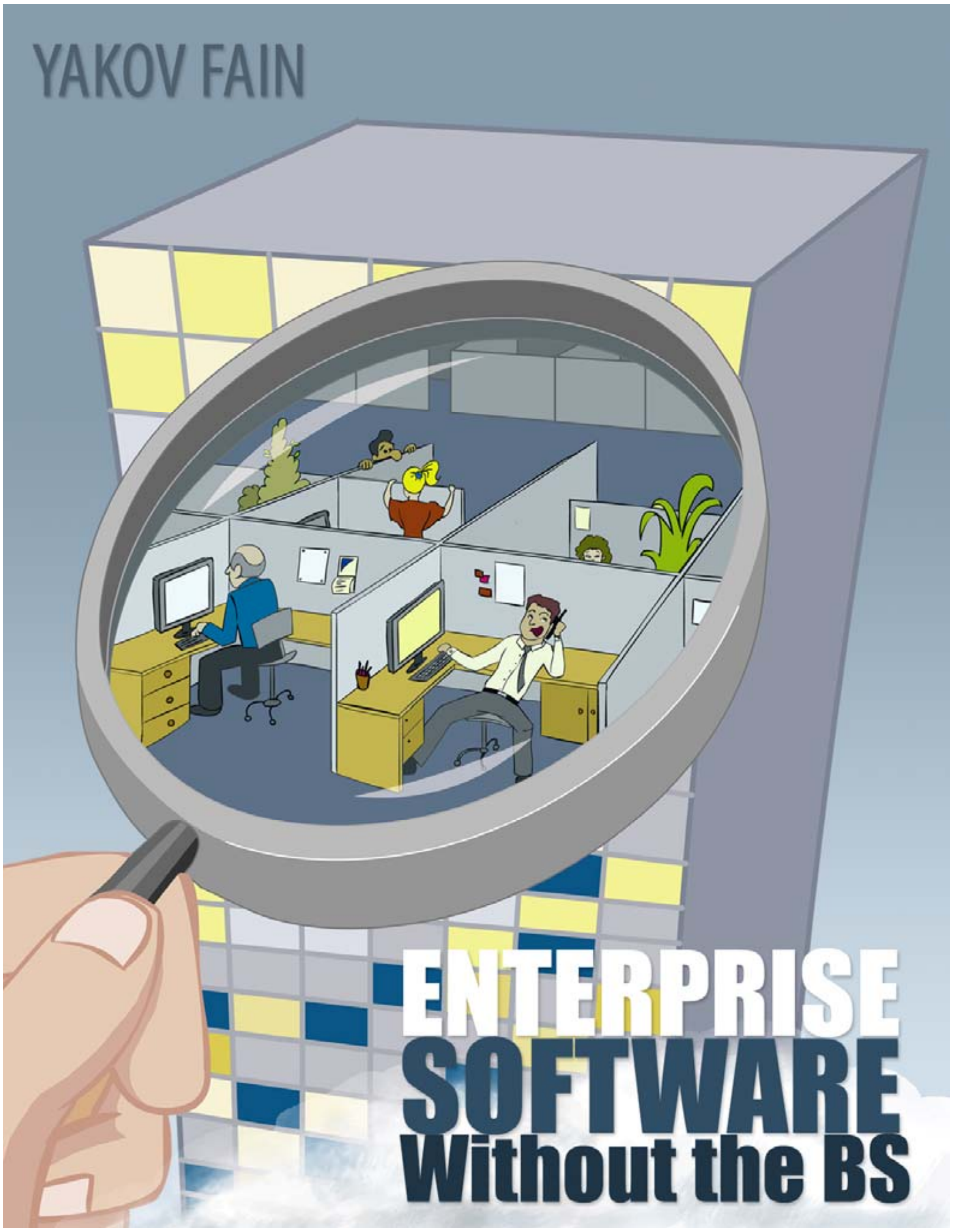
YAKOV FAIN

ENTERPRISE SOFTWARE Without the BS



Enterprise Software without the BS

An ongoing politically incorrect e-book

Yakov Fain

Last updated: June 2008

Enterprise Software Without the BS

by Yakov Fain

Cover design and illustrations: Yuri Fain

Editor: Joey Azoulai

April 2008: First Electronic Edition

ISBN: 0-9718439-1-0

Table of Contents

To my lovely teacher,
Dr. Alice S. Koutkova

Acknowledgements

My deepest thanks to Fuat Kircaali and Jeremy Geelan from Sys-Con Media for all their encouragement of my writing over the years.

My deepest thanks to all bloggers that give me something to read, learn, and think about. You guys make my day!

Thanks to my family for not being angry with me for spending endless hours in my office working.

What’s this book about?

Several years ago I was thinking about buying a gas station in my local town. I went to my friend Gregory Zaltsberg, a successful businessman in this field, and asked him, "How do I start a gasoline business?" He gave me simple but wise advice:

You know nothing about gasoline, but know a lot about computers. Keep doing what you're doing. Just do it a little better than others.

I followed his advice and remained in field of Information Technologies, or to be more specific, Enterprise IT.

I was always interested in observing human relations in the IT business:

Why some people are more successful than others.
In which ways are some people a little “better” than others?
Why people fail job interviews?
Will IT outsourcing hurt your career?
What’s a reasonable salary for a person with YOUR skills living in YOUR geographical area?
Are there underpaid or overpaid people?
How often should you change employers?
Do you even want to have an employer or would you rather work as an IT contractor?
Do you want your child to be a programmer?
How to publish your book?
Me coming to America.
What’s one of the main motivations of innovations in the corporate world?

Prior to this one, I've written technical books, which did not make me richer financially, but definitely served my IT career. I do not expect that the book you are about to read will be become New York Times bestseller (actually, I lie – I do expect this otherwise why even bother?). This e-book gives you somewhat different perspective of the day to day life of enterprise software developers.

This is me in my home office, working on this book

I'll be skipping here and there – can't promise you continuity. But since this is an electronic book and is not available in a printed edition, I have the luxury of patching these holes with new materials in the future.

This e-book won’t give you the answers to all your career questions, but it summarizes my observations formed during my 25+ years of wearing different hats in the Enterprise IT. The odds are that you will not agree with some of my observations, or find some of them cynical. But this is how I see things today, in 2008, in the greater New York metropolitan area.

Part 1: Getting into IT

Do you want your child to be a programmer?

I do. When my older son Yuri was a senior in high school, he said that he wanted to study classical animation in college. What would you say to this? My wife (she's also a software developer), and I said, "OK, not everyone has to be a programmer. We already have two in our family".

Now he's graduated with a BFA in classic animation and works for various TV shows and commercial Web projects (he illustrated this book too).

If he had chosen a career in IT, I could have helped him with every step of the way. I know the IT industry inside out; I know the rules of the game; I could have taught him how to write a resume and prepare for the technical job interview; I know how to set work priorities while working on a software development project... But he's a talented kid, who's not interested in learning all this, and we decided to let him do what he likes.

Once in a while I approach Yuri asking if he'd like me to re-train him to be a programmer so he might doubled his income. He rejected. I respect his position… as long as he pays his bills.

My younger son is in eighth grade, and I've had a secret hope that he'd decide to become a programmer, but he already said that sitting in front of the PC doing the same thing all day long is not for him.

I had this conversation with a colleague who is one of the top IT professionals I've ever met. He does not want his kids to become programmers. My colleague's argument was that 10 years from now all programming will be done in India anyway, and there is no reason to send your kid to a Computer Science (CS) school. I absolutely do not agree with this. Animation industry also have outsourcing issues, and my older son had really tough times finding his first job. He had to accept the first job that did not pay any salary. But his friends, who have graduated with CS degrees, had less problems finding well paid jobs right off the college.

In terms of return on investments, I do not think that there are too many professions that would pay annual salary of $50K to fresh graduates of a second-tier college (not to mention how companies like Google lavish these young kids making up for relatively modest pay for most talented software developers). And if you hold BS in Computer Science from one of the well known schools, your first salary will be $70K or more.

No, the low cost labor from India will not change the IT landscape in ten years. In many cases low cost means low quality:

"Yakov, come on, we are paying only$50K to a team lead in Bangalore!"
"Mary, I'd love to check the damage to your budget by the end of the project. Most likely, it'll run a lot longer and cost a lot more than expected."

The real cost of the outsourced projects is the best kept secret. Good programmers in India are already demanding higher rates, and this trend will continue. We'll talk about outsourcing a bit later in this book.

What happened to enrollment in CS and IS

It's not a secret that number of students pursuing Computer Science BS or AS degrees in colleges is on decline. There are ongoing discussions pondering the reasons behind this trend. Usually the following reasons are mentioned:

1. Outsourcing of junior programming jobs

2. Computer programming is not an easy trade to learn.

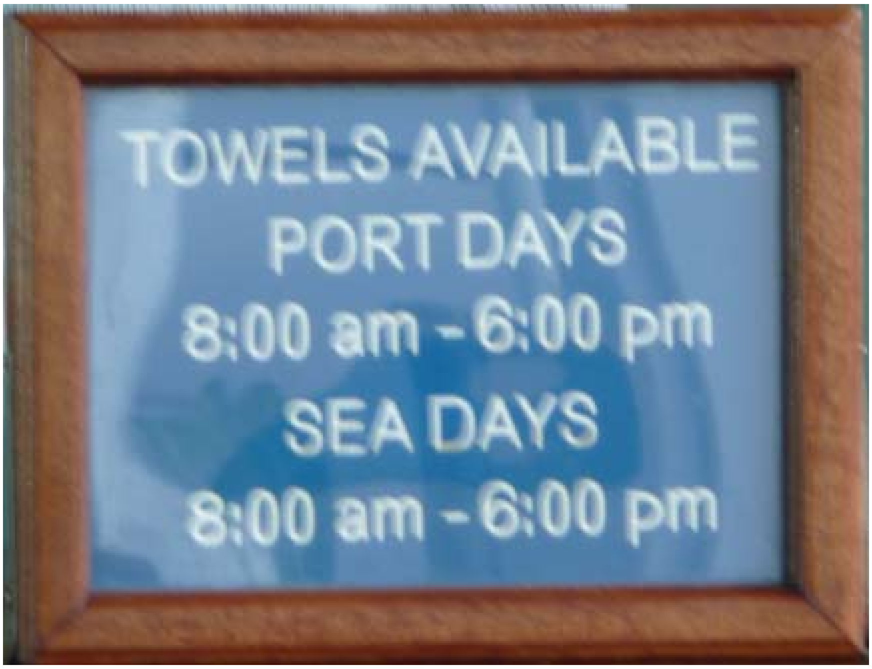
3. If you become a programmer, you'll have to learn new languages and technologies all your life.

4. My child is not too good with Math. In cases like this I sing them the same song, "Majority of business applications do not require any special math skills other than algebra and a simple logic: *if this, do abc; otherwise do xyz*".

But I've changed my tune after one episode... I was waiting for the bus and there were a couple of young people standing by. She was about 25 and he was about 17. She was talking about some event in the past:
"This happened about nine years ago, when I was 16..."
"I can't believe this. You're 25 now? Hold on..." He picked his cell phone and started pressing the buttons. I thought he was going to call another witness of this event for confirmation. Boy, I was wrong! He was using his cell phone's calculator to subtract 9 from 25!

Now if someone asks me if their 18 years old kid should pick CS major, I give them the following advice, "Ask you kid to subtract 9 from 25. If it'll take him/her more than ten seconds, they should pick another major".

Here's another test for your young wannabe programmer. I took this photo on the cruise ship Adventure of the Seas of Royal Caribbean line. Show this photo to your kid and ask him/her what's wrong with this sign.



If you don't hear the right answer within a minute, consider other career for your offspring.

If the USA universities wanted to increase the number of CS/IS students, they should invite more kids from China, Russia and India, which are still good at arithmetic, at least today.

We keep complaining that in American colleges enrollment in Computer Science colleges is on decline. At least we are complaining, but the College Board of U.S. Teachers is simply making the things worse. Just read this article in Washington Post: http://www.washingtonpost.com/wp-dyn/content/article/2008/04/03/AR2008040303925.html .

Now, high school kids will be getting even less chances to enroll in the advanced placement Computer Science programs. American College Board has eliminated Computer Science AB program in schools, since it has low enrollment. Latin Literature, French Literature, and Italian are as unpopular as Computer Science.

And the College Board just finds the easy way out, "If kids do not want it, we'll kill it". Sure, it's a lot easier than making it more appealing to students. In the unlikely event if your elementary school kid will decide to pursue IT career in the future, you'll have no other choice but sending him/her to Bangalore for studying Computer Science.

Still, the scare of outsourcing goes down, programmers in the USA sooner or later find jobs, and colleges slowly but surely revamp their programs to accommodate the needs of the business IT. Two years degrees in information technology make it easier to get into IT field.

Again, the topic of outsourcing deserves more attention and I'll offer my politically incorrect view a bit later in this book.

Out of college: the catch-22 situation

OK. You've graduated with a major in computer science or informational technology, and now face the challenges of the job market.

> What should my resume look like?
> How do I land that first job?
> Should I be picky about my first employer?
> Should I work as a fulltime employee or an independent contractor, and how do I compare the salary of an employee to contractor rates?

Let's start by discussing that elusive first job.

You spent all these years studying and now you're out of college just to find out that it's very difficult to find your first job. You send out a nice (from your viewpoint) résumé,, but you find that employers won't hire you because you lack industry experience. that you don't have industry experience which is why they can't hire you.

You can't get a job without experience, and you can't get experience without a job.

This is the catch twenty-two situation, so you need to have something on your résumé that makes it stand out.

Some people find an easy way out – they lie on their résumés about their past employers. This is bad. It also overcomplicates their first year of employment. Because of this lie they are unable to ask questions to uphold the statements from your resume. Most likely that the person who hired "a liar" knew that s/he was lying, but decided to s/he could do the job…

The better choice would be to gain experience through a college internship, or by volunteering for one of many open source projects. You can easily find lots of open source projects – become a contributor in one of them. Join a project to learn how real-world projects are being developed. Learn how they are set up, what are their version controls systems, nightly builds, milestones, and production releases. These projects will definitely add value to your résumé and, more importantly, prepare you for your career!.

How to look for a job. Can you trust online postings?

No matter if you are just an out-of-college kid or an experienced software engineer, at some point in your career you'll be looking for a job using recruiting agencies.

There are so many excellent programmers who can easily prepare or write a program but they have a really tough time when it comes to looking for a job, getting interviews and so on. We'll spend quite some time discussing all sides of the job search process.

Some of you will remember the days of buying that Sunday fat newspaper for the Classifieds section. In my case it was the New York Times. I would get Sunday's paper on Saturday night so that I could prepare and fax my resume and the cover letter to perspective employers as early as possible. Usually, Sunday edition of New York Times had three to four pages of advertisements of jobs for programmers.

Things are different now, and large and small employers and majority of IT job agencies are advertising their jobs online on web sites like monster.com or dice.com. Spend a couple of months on these sites running searches with the same keywords, and you'll realize that the same ads are published online over and over again, which indicates that many employers have long term contracts with these online job listings in order to attract applicants whether or not a position is actually open.

These employers may not have this exact position, but may (or may not) have something similar. If you are desperate, send them your résumé anyway.

Beware of unusually promising advertisements. For example, everybody is offers fifty dollars an hour for a particular job, and all of a sudden you see an ad from an agency that is offering seventy five bucks for a similar job. Try to stay away from these agencies. Most likely, they don’t have these jobs. They just want to populate their database with quality resumes, expecting that seasoned developers will respond to these lucrative openings. The agencies will save the resumes of these developers in their databases till better times.

You can send your resume, but be prepared to start receiving spam emails offering you positions that don’t match your profile, geographical location or expected compensation. Some recruiters will send your resume anywhere in the country trying to make a buck in commissions.

What real estate agents and recruiters have in common

Simply put, what IT recruiting agencies and real estate agents have in common is that they both work for the other side.

When you are buying a piece of real estate, you hire your own (buyer’s) real estate agent, and the seller has his own agent. If you believe that your agent works for you, do not kid yourself. Your agent also works for the seller because the sooner you buy a house the sooner your agent collects a commission. Many agents will serve their own interests before their clients.

Exactly the same things are happening on the job market. Your recruiters will try to put pressure on you and force you to go to a job interview; they will be sending your résumé to the clients even though you may not like these clients or if you don’t want them to send your résumé they will send it anyway. If you’ll get an offer, they’ll push you to accept it.

Recruiters want to close the deal.

Talk to your agents and ask them to not send your résumé to any client without your permission. Most of the agents will assure you that they will definitely do that and that they will let you know or we will not send your résumé anywhere until talking to you but again three out of four agents will send your résumé without talking to you first. Again this is just experience, they don’t have time and they need to respond to the client quickly before they are unavailable and they don’t want to take chances. So you need to find that one out of four agents who is a trustworthy person.

How to pass a technical interview with flying colors

Regardless if the IT job market is hot or not, there are some rules and techniques that can increase your interview success rate.

The process of getting a job consists of three separate tiers; let's call it an **IPO pattern**:

- Getting the **I**nterview
- **P**assing the interview
- Considering the **O**ffer

I can't stress enough how important it is to work on achieving each of these goals s-e-p-a-r-a-t-e-l-y, one step at a time!

Getting the interview

Your résumé is the most important entity of *tier I*. Adjust it for each position you are applying for. (No, I'm not asking you to lie). Make sure it's short and to the point (I've been developing software for more than 25 years and my résumé is only two pages long).

For example, if you are applying for a Java developer’s position, nobody needs to know the details of that 10-year old Visual Basic project. Unless you are Bill Gates, don't even mention your work experience from the 1980s. Keep good notes of each e-mail correspondence of *tier I* and always update your résumé based on the feedback you receive from recruiters or more experienced programmers that you might know.

Also, many people waste the summary section of their resume with some junk like, "*I'm looking for a challenging position, which would let me improve my talents.*" What a waste! Instead, elaborate on your relevant skills according to the job requirements.

Say you've been specializing in Java messaging during the last two years, but this job posting requires Web developers. Chances are you've developed Web applications before, so highlight your Web experience in the summary section. The same day you may be responding to another ad requesting applicants who know Java messaging, modify your summary section accordingly.

Do not be lazy, work with your resume.

Passing the interview

Remember, your interviewer has a difficult task: he needs to assess your technical skills within 30-60 minutes, so help him! Try to get as many technical details about the job as possible from your recruiter. If the position you are applying for requires knowledge of Java sockets, research working with non-blocking sockets. If they're into multi-threading, learn what the concurrent package is about.

Do your homework and prepare a talk on some interesting and challenging technical problems you might have experienced in one of your college or real-world projects. If you haven't had any super complex projects, just pick a topic from one of multiple online programmers' forums and research it!

For example, if you have prepared a talk on the internals of Java garbage collector, don't leave the interview without talking about this. Even if the interviewer doesn't ask you about GC, do your best to steer the conversation your way. Interviewers will be happy because they won't need to think what to ask next, and you're happy because you've had a chance to talk about a well-prepared subject.

If you're a junior developer, spend some time answering the multiple-choice type of tests that are usually required for certification exams. You don't need to get certified, but all these books and online mock tests will help you pass similar tests offered by some job agencies. Find some sample interview questions online.

Several years ago I've published a set of technical questions for Java developers applying for jobs. This was the most read article I've published. (http://java.sys-con.com/read/48839.htm). More than 600,000 people have read this article. But I urge you to read the comments for this article at http://java.sys-con.com/read/48839_f.htm – it gives me goose bumps. You do not have to be a Java developer to get the picture. India is demanding interview questions. And they want them now! They are actively preparing themselves for technical job interviews.

A technical interview is a game with known rules, in many cases the interviewers are not prepared to run the interviews. They just go by the list of questions. Some interviewees take advantage of this and just spend some time studying just 101-type courses, and then spending most of their efforts on memorizing questions and answers for technical interviews. Believe it or not – in many cases this works. I've gotten thank-you emails from people stating that both interviewee and interviewer were using my list of Java questions during their last interview. Oh well, I hope these half-baked Java programmers are smart enough to keep studying and improve their programming skills on the job.

In your efforts to show-off your technical prowess, don't critique the application architecture of your potential employer - you'll have plenty of chances to provide technical advice after (and if) you're hired, so just focus on getting an offer.

Be energetic during the interview and show your interest in this job. Even if you are a technical guru, don't behave as if you're doing them a favor just by attending this interview. Personality matters. People don't like prima donnas.

One recruiter told me a story about the guy who was interviewing with one of the New York companies. He did a good job during the interviews and was about to leave. On the way out, the receptionist of this company opened the coat closet and asked him, "Which one is yours?"
He smiled to her and said, "Just look for the best one, it'll be mine".

This company did not hire this guy, and I agree with them – personality matters.

As you see, properly leaving the building is also important. As soon as you left the building, get a notepad and take good notes about what just had happened. Don't postpone it till you come home. You may forget some important details, just do it immediately until everything is still fresh. Pay attention to the details work on the questions you've been asked but might have not answered correctly. These question require your attention and more research. Improve your technical skills after each technical interview.

Considering the offer

Tier O: you've got an offer! Now think hard if you want to accept the offer or turn it down. You're in the best position to evaluate an offer, not when your employer decides to let you go or your contract ends, but when you have a stable job, the sky is blue, and the grass is green. A bad offer always sounds better under the pressure of unpaid bills. Have I ever mentioned that you should look for a new job not when your employer decides to let you go or your contract ends, but when you have a stable job, the sky is blue, and the grass is green? This gives you a tremendous advantage: you can consider the offer without being under pressure of unpaid bills.

Don't accept an offer just because the new job pays an extra $5,000 a year, which translates into less than $300 a month after taxes. But do accept the offer that will give you a chance to work with interesting technologies or business applications even if the new job won't pay you an extra dime. Take charge of your career and actively build it the way you want.

Interviewing enterprise developers

Now let’s take a peek at the interviewing process from the other side of the fence as the chances are that after accepting the offer, you’ll be asked to interview other job applicants.
When the job market is healthy, major online job search engines show thousands of openings, and people are competing for these jobs. In 1996, skilled Java developers are just as popular as Visual Basic or PowerBuilder developers. There is a major difference though - back then, client/server developers could make a decent living by mastering one front-end tool and any major relational DBMS. These days seasoned developer has to know about 10 different tools or technologies to find a good job and feel relatively secure for a couple of years.

Over the last couple of years, I've been interviewing lots of Java developers – they are in demand again. But current job requirements, people, and resumes of Java developers have changed quite a bit, and this is what I've noticed:

- People do not call themselves Java developers or programmer-analysts anymore - most of them prefer the title of Java architect. Unfortunately, only some of them really understand how J2EE components operate and can suggest some design solutions.

- Job applicants are more senior, and I barely see any college graduates or junior programmers in the market. Many of the junior positions are being outsourced and the number of graduates with computer science degrees has declined over the past several years.

- Having software certification does not make your resume stand out. Actually, if a résumé starts with a list of certifications, most likely it's a beginner. I'm not against certifications as they help you to learn the language or a tool, and show that you are willing and can study. But the fact that you have a certificate doesn't mean that you're a skilled professional.

- With introduction of middle-tier object-relational mapping frameworks like Hibernate, many people don't even bother learning how exactly how the database management systems work and how to write a well performing SQL query - they just pass a SQL statement to some wrapper class created by local architects and get the result sent back.

- I see a new breed of Java architects who used to be project managers. These people usually know their business really well, can talk about application servers, messaging and clusters, and capacity planning, but often fall short on Java technical questions.

- Job requirements are longer these days and recruiting companies don't even want to submit your résumé to the client if you have "only" 8 out of 10 required skills. As a matter of fact, recruiters screen candidates a lot better now.

- Be prepared to pass at least four interviews to get hired. While back in 1999 two good interviews would be enough, in 2001 it was very difficult to even get an interview let alone a job!

What does a good enterprise Java developer have to know in addition to understanding the difference between abstract classes and interfaces? Usually employers are looking for people with at least 10 of the following skills: Java servlets, JSP, Struts or a similar framework, EJB, JMS, any commercial message-oriented middleware, JDBC, JNDI, HTML, XML, Spring, Hibernate, Ant, SQL, one of the major application servers, a couple of relational database management systems, any UML modeling tool, several design patterns (at least a Singleton!), and familiarity with Unix.

Understanding why a particular J2EE component is being used in your project is equally important. If the interviewer asks you, "Why did you use EJB in this project?" please do not answer, "This decision was made before I joined the project." Have your own opinion and explain why you think it was a good or bad choice for this particular project.

I keep hearing the "horror stories" about questions some people get during interviews. In my opinion, the interviewers should ask more open-ended questions about the applicant's prior experience, going into technical details when appropriate. I don't think it's fair to ask a person to write a Java program processing a binary tree or implementing a finite state machine. These are the things that can be looked up online or in books when needed.

Good knowledge of the business terminology of your potential employer is also important. I'm not sure about Silicon Valley or Europe, but here in New York just being a techie may not be good enough to get a senior job.

For example, if you're applying for a Java position in a financial brokerage company and don't know what a short sale is, this may be a showstopper. If you are a senior developer, you should be able to hit the ground running… Try to find out from your recruiter as many details as possible about the business of your potential employer. Do your homework, and you'll get the job! They are desperately looking for good programmers and you can be one of them.

Give a second chance

Today you are a job applicant, and tomorrow you'll be asked to interview people. Are you ready to do this? There are periods when the job market is bad (the seller's market), and after a while it becomes hot (the buyer's market). Employers should adjust their interviewing techniques accordingly. In a situation when development managers can't find the right candidates and can't staff their new projects, they need to apply a different technique, which I call "A Second Chance". Let me explain it by example. I'll use some topics from Java, but you can easily identify similar cases in any programming language.

After the interview, the project manager may get one of these feedbacks on the job applicant:

"He's a good guy, knows Java and Web applications, but he never worked with Struts. We're going to pass on him".

"She has many years in Java development, and even had a chance to work with Swing, but the fact that she does not know the difference between the methods *invokeLater()* and *invokeAndWait()* is a clear sign that she is not the right person for a Swing project".

Now, tell me this, if a good Java programmer does not know Struts, how long do you think it'll take him to learn this particular framework? A week? Two weeks?

Or how long does it take to people living in the Google era to find the difference between *invokeLater()* and *invokeAndWait()*? An hour? Ten minutes?

Do not let good people go.

Do not lose good people, give them a second chance. Call the job applicant after the interview, and tell him/her something like this, "Joe, I see that you have a solid knowledge of Java and understand how to develop Web applications. We'd really like to hire you, but since our project heavily relies on Struts, we want to make sure that learning Struts is not a big deal for you. Please spend a week learning Struts in your own spare time and call us. We'll spend 15 minutes talking over the phone on this subject, and after that we may extend you an offer." This is simple and efficient way of bringing good people on board.

Joel Spolsky has written an excellent article on interviewing programmers (http://www.joelonsoftware.com/articles/GuerrillaInterviewing3.html). Unfortunately his technique is not always applicable in large organizations, but his message is clear: hire good people. You can teach a good person some new programming techniques, but it's not easy to make a technical geek a good person.

And finally, if you can't find the right person to hire as an employee, bring a consultant on board with specific technical skills. Even if it's more expensive, it's a short engagement.

Employment is somewhat similar to a marriage. You don't get married until you find the right second half, right? But looking for the one-and-only should not stop you from seeing other men or women.

And he hung up during the interview

I got this email from a strong Java developer I've known for years. He writes, "Today was the first time in my life that I hung up the phone during a job interview". I asked him why. The following is his reply:

The interviewer asked me how would I timeout a URLConnection. I said that I would simply send a ping once a second and if we did not receive a ping we considered the connection dead.

Again,The interviewer asked me how would I timeout a URLConnection. I said that we sent a ping once a second and if we did not receive a ping we considered the connection dead and notified the user and blah, blah, blah.

The guy said that you should specify the timeout a URLConnection and asked how would I do it. I said that.
I am not sure there is a way to specify timeout in URLconnection since originally in Java all I/O was blocked.
The guy asked me how would I timeout a URLConnection again! I said that I would do the timeout on the software level; that I am not aware of any other way to do it, and that in the very end I am really telling him how the system works and if he can give me a hint to what he wants I'll be glad to answer.

The guy asked me how would I timeout a URLConnection. I am not joking, he asked me again. I said - we did it using ping message if we did not receive the ping within 2 seconds we considered the connection dead.

The guy asked me how would I timeout a URLConnection. I hung up.

I guess, the interviewer went by someone's list of questions and answers, and was expecting the following answer "Just close the underlying I/O stream and the connection will be timed out". His list did not include the ping alternative. There are morons working everywhere, and this one happened to be working in a prestigious Wall Street firm. However, the job applicant's evaluation of the potential employer is irrelevant at this stage of the game.

During the interviewing phase, you have the only goal – pass the interview successfully.

In this particular scenario, the best outcome would be receiving an offer, and have the luxury rejecting it.

Your first Employer

Does it matter who is your first employer?

In my opinion it doesn't matter at all because most likely you don't know what you want to do with your career, you don't know what your strengths and weaknesses are. Sometimes, a bad employer can help you decided what you do and don't like better than a good one.

Barring any heavy financial obligation, your first employer and your first salary do not matter as much as what you take from the experience. You should be working with modern technologies, learning how to collaborate with teams of programmers, and learn how to communicate with non-technical people. Obtaining these skills is the most important goal for your first years of employment.

In the beginning of your career you should switch jobs more often. In general, you shouldn't work for any employer for more than five years because your technical skills become rusty, you become complacent, and might lose the motivation to learn new skills because the majority of your time is spent not on writing code, but on resolving issues specific to this particular project.

Make a move after five years if not sooner. Mastering communication skills is as important as mastering hot programming languages. Learn how to get what you need, who to talk to, how to talk the talk, and how to manage your time.

There are many great programmers who just cannot deliver on time. They get carried away, they try to solve complicated and challenging problems, and when the project time is up, they may be behind even with simple tasks.

When your manager asks you to do something by Friday it has to be done by Friday. Don't try to give her more than she asked for. Deliver exactly what she's asked you to do. You manager is responsible for the success of the project; please do not jeopardize it.

And do not forget about teamwork. You need to get along with other people that were born and raised in different countries, who could have different skill sets, different communication habits, and so on. Twenty percent of a typical team consists of strong people, while the other 80% are people with mid or entry level skills. You have to learn how to work with all the people in your team.

An ideal team consists of people who do something better than you.

Some of them are better coders, some have better communication skills, some can generate great ideas daily, some are better analysts and can turn a complex task into a set of simple steps. Stay in the team till there is something to learn. If you feel that there is nothing to learn, start looking for a better team.

To be a good team member, you need to do something better than others too.

At this point, I'd like to sprinkle some historical facts for those who are just entering the world of coding.

An unofficial history of programming – '95 - present

I want back into the '90s. Seriously. Twelve years ago I didn't know Java; I'd been using PowerBuilder and was able to program pretty much everything in this RAD object-oriented tool. To find a job back then, all I needed on my résumé was PowerBuilder, a single framework (PFC), and SQL. With these skills I could have created a prototype of a rich Create-Read-Update-Delete (CRUD) client/server application in a couple of days. However, that was the sunset of the client/server era.

While making the deployment of client software easier (a Web browser is all you need), the Web 1.0 pushed user-facing applications years back. Just look at these ugly screens: several plain text boxes, a dropdown, and a trivial HTML table. Mainframe dumb terminals had black screens with green letters, but the interaction with the big iron was super fast. The Web offered a white background with black letters and poor performance. But the entire world was so happy with this new way of accessing the wealth of data and tons of opportunities in e-commerce, that people were willing to put up with some minor inconveniences.

In 1994, the Gang of Four released a famous book called "Design Patterns: Elements of Reusable Object-Oriented Software". This book was the first step in turning programming from an art to a trade. Singleton, MVC, Factories, Data Transfer Objects... just pick up the proper design pattern(s), and your code will look as if it was written by an expert. Don't forget to comment your programs explaining which design patterns were used in your code. Still, there is a small number of programmers who get by without pattern programming, but they'll be extinct soon.

SQL was in favor in the '90s. People knew how to delete duplicates from a database table by applying such SQL clauses as group by and having. How many people have read the book by Joe Celko, "SQL for Smarties"? Let me put it another way. How many people know what SQL is? Why bother, a persistent framework like Hibernate will let me map class attributes to the database table columns. How nice...I'm drowning in XML now. Let's not jump ahead though; mankind did not know Hibernate or XML back then.

The Java programming language was born. It became visible as a language for creating applets, but its uses for the desktop were quickly abandoned for the server side. It took Sun more than ten years to realize that desktop programming is also important, and that is was time to create a Swing-based RAD tool.

The end of the last century can be called the Gold Rush of Programming. People started to spread the fear of Y2K issues. Since the dates (years) were stored as two digits, some nuclear explosion or a less serious disaster was expected on January 1, 2000. For example, performing year subtraction "'00-95" would give you a negative number. Get it? Lots of people quickly became programmers with the noble mission of saving mankind. Lots of IT managers quickly climbed the corporate ladder working on this mission.

In the beginning of the new century, XML became popular. Yes, it was a nice way to describe data, but at the same time it was too heavy. It did not manage to kill the CSV format - the hype is over - but it did find its use in a variety of applications.

Microsoft came out with .NET platform, which became a direct competitor of J2EE. Today, most of the enterprise software development is done using these two mainstream technologies.

Another important trend of this century is the spread of open source software. In the past, vendors used to sell software licenses, but now many of them give the software away for free and sell services instead: "The documentation of our open source product may be poor, but no worries, we'll be happy to help you with our great tool for an extra fee." There is nothing wrong with it – software developers have to eat too.

Many of the six million Java programmers won't even consider using a tool or software component if it's not free.

What are the latest notable trends? Let me throw in a couple of buzzwords:

AJAX is a self-proclaimed savior of Web applications. You enter a letter in an HTML search text field, and the results come back without refreshing the entire page. While AJAX is the right solution for prettifying your Web pages, some enterprises got into a risky business of developing large-scale applications using AJAX. Big mistake. Huge. I spelled it out in this blog: http://flexblog.faratasystems.com/?p=163. But since there is a demand, many vendors are making their tools AJAX-enabled.

Meanwhile Java developers go crazy because of this orgy of 50+ Web frameworks that do the same thing as the Struts framework. The AJAX developers also “enjoy’ a variety of more than 100 frameworks that do the same thing. James Gosling, a creator of Java was interviewed at the JavaPolis conference in December of 2007, and someone asked him, “Why are there so many different Web frameworks in the world?” He gave a funny answer; “There are too many creative people in the world. You should all get stupider…Some of it is that people just like to have fun”.

On the other hand, .Net developers mainly deal with one vendor – Microsoft, which produces quality software for writing software, and a wealth of professionally written documentation. For years, Microsoft enjoyed the status of the number one software development company, but this new kid on the block Google quickly evolved from just another search engine company to the most prestigious employer in the software world. They offer free software as service for the masses while making money on selling online advertisements.

The very existence of Google bothers Microsoft. These two companies make tons of money by using absolutely different business models (selling licenses vs. selling ads), but today’s cool-factor is on Google’s side.

During the last three to four years, lots of enterprise mission-critical systems were moved from the Unix to the Linux platform, and this trend will continue.

The Ruby on Rails library is heavily promoted by a group of enthusiasts that create Web applications. They are pushing the idea of “conventions over configurations”.   It's not likely that Ruby will become a commercial programming language, but it has a decent number of fans. It may remain yet another good language such as Lisp or Smalltalk.

Rich Internet Applications are in favor now; I'm talking about fat clients here. The major players are Adobe Flex and Microsoft Silverlight. Sun Microsystems is trying to enter this market with a new language called JavaFX. This is an interesting field to be in today.

There is something called Web 2.0. The term has been never defined but people seem to like it. In my opinion , it's just about giving more control to the users of the Web applications. More control – higher number (3.0, 4.0… N.0)

What about us programmers? We have to keep learning more and more buzzwords/tools/frameworks/languages to become senior software developers...oops, I meant to say architects. Why not developers? Because only architects can possibly figure out how to put all these unrelated pieces together.

I want back in the '90s. Seriously.

Part 2. Living in IT

Some recommendations to young developers

Once in a while experience software developers post advices for rookies. For example, these are a couple of recent sets:

http://tech.puredanger.com/2008/04/20/advice-to-a-young-developer/

http://dsoguy.blogspot.com/2008/04/more-advice-to-young-developer.html

Giving advices is easier than following them, and here's my set.

1. Being a software developer does not mean sitting all day long with your ear buds creating cools widgets for the Web. Coding of cool things will take less than 25% of your time. The rest of the time you'll spend communicating to other people. Be good at it.

2. You know how to do things right. Guess what, other people may have different opinions, and they might be right. Listen, learn, and adjust.

3.While you solutions may have technical merits, there can be other reasons for not doing it "your way". It's OK. Do not get frustrated.

4. These 40+ years old farts in the cubicles may not have accounts on MyFace, FaceBook, or Twitter. But that can't be too dumb if they've managed to survive in IT for 15-20 years. May be they know something useful for you to learn?

5. Slow down. You are typing too fast, but the most worn out button on your keyboard is Backspace. Think first, then type.

6. You feel underpaid and being taken advantage of? Hit the job market. You might be surprised – for some reason no one else want to hire you. This may give you a piece of mind and ring a bell – it's time to study again.

7. You must purchase and read at least five technical books a year. Googling for specific solutions is fine, but you need some more fundamental knowledge. Do not save money on books.

8. Use every opportunity to attend technical conferences. It'll give you a better sense of where the industry goes so you can invest your time into learning what's needed today.

9. Start blogging on technical discoveries that you made recently. Even if these little findings may not sound too important for you, they might be life savers for someone. Blogging will improve your technical writing skills and will give you some visibility.

10. Get your Master Degree before you got married. It does not have to be MBA – get it in the field you've always dreamed of. When your significant other will come into picture, going back to school becomes a lot more difficult. "Honey, am I right or am I right?"

Managing your software development career

Don't wait until you need another job to look for other opportunities. Just check the job postings on a regular basis and respond to them. You do not have to leave your current job immediately but, it's worthwhile to get a good understanding where the job market is moving, and prepare yourself for a rainy day.

I am participating in different technical forums on the Internet, and some people add their signatures right below the postings. On one Java forum, I noticed two people who would always help other people with questions. One of them signed their postings, "John Smith," followed by the statement that he's accepting new projects. The other person signed their postings, "Mary Lou, unemployed programmer", which is a huge mistake. Even though both signatures tell me that both of them are looking for a job, I'd rather deal with the first one who is more positive.

When to take a new job

Why switch jobs when the one you've got "ain't broke"? A small bump in pay - by American standards say, $5,000 – is not a good enough reason to leave a job. Take a new job for the chance to work with cutting edge technologies. Keep the long view in mind!

Rules of resignation

Resigning is a very sensitive process, and there are some rules you should be aware of.

1. *Don't resign just because you are angry with your boss.*

Immediate resignation is not a good answer if the boss was rude, or there was some other conflict at work. Start looking for a job. Don't resign unless you've found a new job or have a substantial amount of cash in your bank account.

2. *Announcing the resignation*

Give two notices -- one verbal, and the other a written resignation letter or an email. Importantly, the resignation should be short. You shouldn't write a long letter explaining that you are unhappy, your contribution to the success of the project was not appreciated, etc.. Just write one sentence, for example,

"This is a resignation notice, and my last day at work will be October 15, 2008. Thank you for a great experience and a great time, but I've gotten an offer that I cannot reject".

This is more than enough.

Don't explain your reasons, and never complain about anything when your boss or a person from the human resources will be talking to you about your resignation. They may ask, "What did we do wrong? Could you give us some advice so we'd do better in the future?"

Do not fall into this trap. They won' t accept their wrong doings anyway, but your advices will irritate them. Do not forget that you may need to call this person in a month (after resigning) asking for references. Don't teach them how to run their business. Mind your own business.

3. *Never accept counter-offers.*

This is a golden rule. Even though it might sound very flattering that now they appreciate your work and are matching your new job offer, you need to understand that your employer now realizes that you are ready to move on.

Most likely, he is giving you a counter offer just to buy some time to find a replacement for you (and so you can transfer knowledge to this new person). But as soon as the rainy day comes, and the firm won't be doing that great and your boss will need to make a choice, they will cut you loose. So just leave and don't take any counter offers.

Don't forget about the COBRA program, which is federal law stating that you can continue getting your medical benefits for up to eighteen months, which maybe important because your new employer may not give you medical insurance right away; you may need to work say for three months until you get such coverage.

4. *Don't post negative blogs about the company you quit*

It's not nice, really. If you don't like your firm, just leave it. When I read a blogger badmouthing a former employer, I just lose any respect for the blogger. It's not professional, and besides, I've heard that some companies have started suing their former employees about these bad postings online.

Here's an example of how a person is trying to boycott his former employer:
http://www.hiveminds.co.uk/node/3751
Here we can only hear the opinion of the angry blogger, but without

knowing the employer’s story it’s hard to judge who’s right here. But I can definitely tell you that I would not hire this blogger even he’s the best developer for the job at hands.

Who are these IT contractors, anyway?

Contractors are people who work for a particular company for a short period of time, but they are not employees of this company, hence they do not receive salary and benefits (paid vacation, medical insurance, etc.) from this company.

Contractors work for money. Period. They do not have any other objectives such as making a career, earning a new title, or good retirement package. They work for a firm mostly because they like the pay rate, and in some cases, they are interested in learning technologies that are used in said firm. Can you have peace of mind if you spend all day working on something you do not really enjoy? In a perfect world, people will work on very interesting and well paid jobs. But since our world is not perfect, work on the interesting stuff after hours for free. This hobby may turn into a paid job in the future.

Most contractors work with their clients through another firm that was lucky to get on the client's preferred vendors list. Such firms are often referred as *pimps,* which does not mean that contractors do not like them. Over the years, experienced contractors create their private lists of reputable pimps and maintain friendly relations with them.

In terms of paying taxes, contractors (in the USA) can work in various ways , for example on W2 form, which means that the pimp pays your taxes: you'll receive the net income according to your tax brackets. Creating your own corporation can be more rewarding in terms of taxes, and in the industry jargon , this way of working is called "corp-to-corp". There are some other ways of working, for example on 1099 form, when you are a sole proprietor (a.k.a. independent contractor). Sometimes, these independent contractors are called freelancers.

Here's another term: "contract-to-hire". This is also called "try-and-buy": sometimes the client company wants to try you as a contractor first, and then convert you to an employee if they like you. If this is does not meet your career objective, either do not take this contract, or negotiate upfront if working as a contractor only is an option.

People use other forms of businesses too, i.e. Limited Liability Company or L.L.C., but this discussion is out of the scope of this book.

Some are of the opinion that working directly with the client is better than going through the pimp. This is not necessarily true. For example, if your direct client goes out of business, you'll never receive your last check(s). I've been in such a situation myself, and my lawyer said, "You can't get blood from the stone". In this regard, pimps are more reliable parties to deal with.

How much does a client pay to your pimp for your services? Typically, your pimp enjoys a 15-30% markup (this number is a lot higher in offshoring situations). The larger the pimp, the higher this number. Is this a reasonable amount? Try not to worry about it. Your pay rate should be your only concern. Do not like it? Try to find a better rate somewhere else. Can't do? Sit down and shut up. Welcome to capitalism.

Hire a professional accountant to do your taxes. Never try to save money by filing your taxes with the help of some inexpensive software. Reputable accountants know how to save your money based on the loopholes in current tax laws in your geographical area.

Is it ethical to terminate your contract before it officially ends? Yes it is, but play by the rules; always give a two-week notice to your client. The client may not be as nice to you and can kick you out without any warning at any moment. This is not typical behavior, but it happens.

When I was signing my very first contract with a large Wall Street firm, I did not like the clause that I'd have to give a notice but the client could terminate the contract at any moment without explaining the reason. I tried to argue, and the client's manager pointed at their (employees?) on the open-space floor and said, "Half of these people are lawyers. Do you think you have a chance to win if something goes wrong? No way. If everything goes well, we'll always give you an advanced notice". This gave me piece of mind and I signed the contract.

If you are a contractor, your relations with the client's team are usually pretty straightforward. The project manager likes you, because you help him to complete the project in time, which means that you help is his/her career. Your peers who work on this project as an employees might envy you, because they believe that you earn a lot more than them, without having to put up with the corporate politics, et al. They may not consider that you may have non-billable time between the projects, have to market yourself, pay premium for the medical coverage. But they may respect you for your expertise as a sort of a necessary evil.

Why do companies hire these expensive contractors? The short answer is they need a person with a specific set of skill for a short period of time. I'll explain it in more details a little later in the book.

To get an idea about contractors' rates, check out the Web site realrates.com.

Here's a good article by Steve Friedl "So you want to be a consultant?". Please note the section "You must give the customer The Warm Fuzzy Feeling ". Steve is absolutely right – in consulting, human relations often mean more that technical abilities of a contractor. Hence, *"they would rather pay her to "figure something out" than hire somebody else who already knew how"*.

Chad Myers shares his ten rules of consulting in his blog. His main message is "The customer is always right". In my opinion, Chad suggest a consultant to bend too much, but in general he's right. If you are hired as a consultant, you'd rather be a source of solutions, than pointing out customer's problems.

Work as an employee or a contractor?

Should you work as a full time employee or as a contractor? It's hard to give advice here, but if you are working as an employee and thinking of becoming a contractor, have a conversation with yourself:

"Why do I want to become a contractor?"
If the answer is "I've heard that contractors earn more money",
Ask yourself, "Am I ready to maintain my marketability by spending lots of extra hours to keep my skills up to date?"

If the answer is "Maybe", consider just getting another part-time job (deliver pizza or something). This will bring you extra income without the need to lose the stability that permanent employment gives you, and you do not need to work so hard on improving your skill level.

But if you have that burning desire to be your own boss, try it out, but first consider various other options of earning income.

Just to recap, you can work as a full time employee of a consulting company which acts as a middleman between you and their client. You can work as a contractor "on W2". You can work as an incorporated contractor or as a freelancer (self proprietor). A full time employee is pretty much as simple and clear as a W2-contractor getting paid based on an hourly or daily rate but all the taxes are withdrawn by the company who pays you, you just get a net "after-tax" pay.

You are an incorporated contractor, if you've opened a corporation, which is pretty easy in the USA. The client company will pay your corporation or as we say "on a corp-to-corp basis". They will pay you a gross amount of money, and it's going to be your responsibility to keep track of your expenses and pay taxes. In this case most likely you will still work at the same place every day; you will have your own cubicle, and you will be working the same hours as the other employees in this company.

Freelancers, the minority, usually work on short assignments. They may not even go to work every day, but rather work on a well defined assignment, for example, developing a particular software component or run training classes around the country. To become a freelancer, you have to have good professional reputation and connections.

So, which way to go? Which form of employment is good for you?

These are some of the reasons to work as a contractor:

- Your title is not important to you.
- You'd like to have a chance to work with different technologies.
- You like always learning and you improve your skills all the time.
- You hate corporate politics.
- You like meeting new people and working in different environments.

Work as an employee if:

- Your title is important to you.
- You'd like a chance to have a nice office with a window in a place other than your own house.
- You want to have a plate on the door with your name on it, but you do not feel like ordering such a plate from a mail-order catalogs.
- Job interviews are stressful for you.
- You or somebody in your family needs good medical coverage.
- there are just no contract jobs where you live.
- You are one of the first employees in the startup of the company and you have some options that you can't cash in just yet.
- Your spouse already works as a contractor and someone has to ensure uninterrupted steady income.

Comparing the incomes

The next question is how to compare the income of employees and contractors. How to compare an annual salary with hourly rates? Let's go by example.

Mary is single, she works full time, her annual salary is $70K, she has two weeks of paid vacation, ten paid holidays, five "sick" days and she pays three hundred dollars a month for her medical insurance.

Now she gets a job offer from another firm. It's a one year project on a corp-to-corp basis with pay rate of fifty dollars per hour and no benefits, of course. Will Mary's bank account be merrier if she takes this offer and becomes a contractor? Let's do some math.

Currently, Mary's tax bracket is 33%. So her net annual income is $70K minus thirty three percent, which makes it $46900 and after subtracting $3600 for medical expenses, she takes home **$43300** a year.

What if she'd be working as a contractor on a "corp. to corp. basis"? We'll be using her weekly pay to calculate her annual income. There are 52 weeks in a year, but to compare apples to apples we need to take into an account that Mary-the-employee has two weeks of paid vacation, ten paid holidays and five "sick" days. So we need to subtract five weeks of lost pay if Mary becomes a contractor but wants to enjoy the same number of days off work.

We'll use 47 forty-hour weeks in our calculations of an annual gross income:

$50* 40*47= $94000.

As every U.S. corporation, Mary would need to pay additional social security tax (7.5%), but by claiming some of her expenses as business-related, Mary legally reduces her tax bracket to 30% and her net income becomes

$94000* 0.7= $65800

If she need to buy a medical insurance, she won't get a $300-a-month deal. Mary-the-contractor will need to pay $800 a month so let's subtract another $9600 dollars to calculate the net income:

$65800-$9600=**$56200**

Mary-the-employee's net income $43300, which is less than the Mary-the-contractor's $56200.

I gave you an example assuming that Mary is going to be contracted for the whole year.. This may not be the case, and she may have some non-billable time between contracts, which means lost gross income of $2000 a week, which translates into $1400 of lost net weekly income. Six weeks of downtime would lower her net income to

$56200-$8400=**$45800**

If you do not see any flaws in my calculations we can say that from the financial point of view, it's better to work as a $50-per-hour contractor than a $70K-a-year employee. I did not take into consideration other perks that your employer may offer like matching contribution for your pension plan, laundry on premises and discounted tickets for the second-tier Broadway shows.

Working as an employee or a consultant can often become a subject of heated discussions. Jim McGovern wrote a blog titled "Are you a consultant": http://duckdown.blogspot.com/2007/09/are-you-consultant.html#links . Jim is employed as an enterprise architect by a major insurance firm, and he is a popular blogger. We've co-authored a book together, and I can assure you that he definitely knows what's happening in the real enterprise world and his insights are often interesting and thought provoking.

I'll just take Jim's observations about consultants (a.k.a. contractors) and will respond with similar observation about employees:

You work very odd hours. It's difficult to maintain a relationship or a family. You are paid a lot of money to keep your client happy. You are paid well but your pimp gets most of the money. You spend a majority of your time in a hotel room.

You work very odd hours without being paid for all these endless evenings you spent in the office to keep your manager happy. You salary is OK, but you've been brain washed that your total compensation package is twice as big, because you are getting extra perks like 401K plan with no or minimum contribution from your employer, free car wash, laundry on premises, gym that you do not use and discounted tickets to the Broadway shows that you do not like. You spend majority of your time at work.

You charge by the hour but your time can be extended for the right price. You are not proud of what you do. Creating fantasies for your clients is rewarded. You have no job satisfaction. If a client beats you up, the pimp just *sends you to another client. People ask you, "What do you do?" and you can't explain it.*

You do not charge by the hour, hence it's assumed that your time is not worth anything. "By the way, Joe, can you join the meeting at 6AM tomorrow so our offshore partners won't need to stay late?" If you do not like what you are doing, you are pretty much stuck, because even if, in theory, you can ask for a transfer within the same organization, you'd better plan to spend the rest of your employment with your current manager. Your only hope is that your manager will get promoted and will be transferred somewhere else. But will the new manager be better?

Your client pays for your hotel room plus your hourly rate. Your client always wants to know how much you charge and what they get for the money.

Your manager knows exactly how much you make, and wants to make sure that he gets as much as possible for the money. He also knows that one can't get blood from a stone, but given your modest salary, he can live with it.

You know the pimp is charging more than you are worth but if the client is foolish enough to pay it's not your problem. When you leave to go see a client, you look great, but return looking like hell (compare your appearance on Monday AM to Friday PM).

You believe that you are underpaid and it bothers you all the time. You do not think that it's fair to work your butt off as you do day in and day out. Compare your appearance on Monday AM and Friday PM. Actually, you may get a call from work on a weekend too.

You are rated on your "performance" in an excruciating ordeal. Even though you get paid the big bucks, it's the client who walks away smiling. The client always thinks your "cut" of your billing rate is higher than it actually is, and in turn, expects miracles from you.

You are rated based on stupid rules invented by someone from the HR department. All year you earn points for good behavior. The highest points are earned if you fill and submit your timesheets with a detailed description of your work on time. If your group actively participates in one of the firm-wide social initiatives, they'll allow you to come on Wednesday wearing jeans. Your manager does not expect miracles from you, but is pleasantly surprised when you deliver one. Hint: save your miracles till October/November when the raise and promotion decisions will be made.

When you deduct your "take" from your billing rate, you constantly wonder if you could get a better deal with another pimp. Every day you wake up and tell yourself, "I'm not going to be doing this stuff the rest of my life."

When they deduct your taxes from your gross income, you constantly wonder if you could get a better deal with another employer. Once in a while a crazy thought strikes you; "Maybe I should try to become a consultant myself?" But then you arrive to your comfy cube, see familiar faces and then say to yourself, "No big deal; I have only 15 years to my retirement. If I control my emotions and maintain good relations with my boss, I'll spend these years without major cataclysms".

So, are you a consultant or a prostitute?

Aren't we all…

Another Brick in the wall

The subject of working as a contractor vs. employee deserves more attention. This time we'll add some great music to make ease the digestion of this material. Put on your headphones, turn on Pink Floyd's The Wall (please…do not tell me you don't have it), and keep reading...

You know by now that software developers earn their living by working either as employees or as temporary contractors. Often, people use the term *consultant* when they're referring to the employment status of a person, but this is just not right, because the word consultant means a subject expert, while the word contractor means a temporary worker and a separate legal entity, which is exactly what consultants are. There is an opinion that permanent employment provides better job security, but let's take a closer look at two former college roommates, Alex and Steve, who graduated from the same college eight years ago.

Alex was always dreaming of being an employee of a large corporation. He knew that he'd be more secure there (*Momma's gonna keep baby cozy and warm*) and was ready to work for such a firm for many years. He found such a job and had to start from scratch learning the rules of the corporate world: your phone conversations may be recorded, a designated person will browse your e-mails, your applications will be protected by a couple of firewalls and DMZ (*Momma won't let anyone dirty get through*). He had been promised regular training classes and business trips to San Francisco to study new Java technologies at the JavaOne conference.

Six months later…

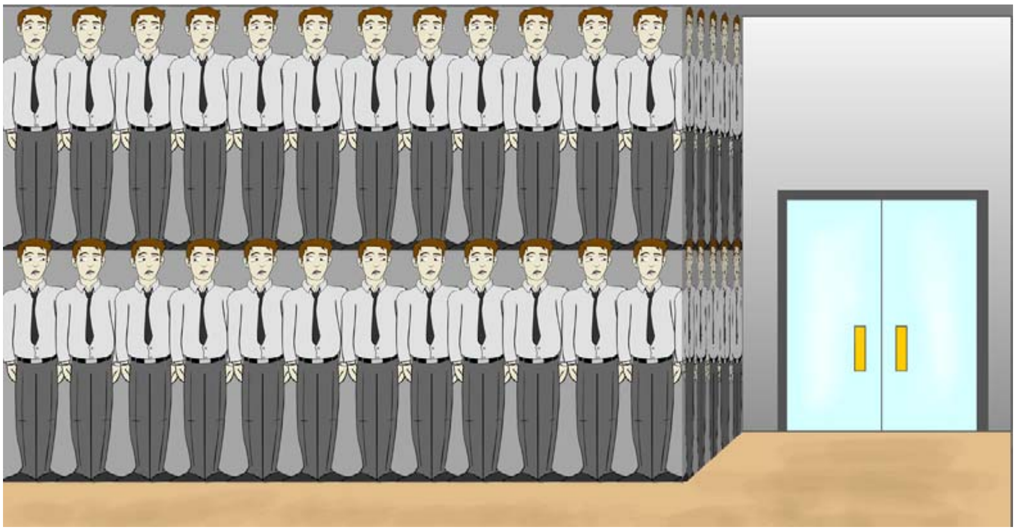
“Sorry, but our training budget is not as good as it used to be and can’t send you to SF (*We don't need no education*), but we have an exciting Six Sigma training coming up, which will greatly help your career, and you may even earn a green belt in a couple of years.” He learned to play politics, and got used to working late hours to meet unrealistic deadlines that were set by some incognito bad person from up above. Alex met all deadlines because bonus time was looming ahead (*If you don't eat yer meat, you can't have any pudding*).

Steve decided to work for himself, so he opened up a one-man company and started his career as a contractor. Even though his contracts were usually long term, Steve always knew that he needed to maintain good technical skills to be prepared for the next technical interview. He was the first to learn Aspect Oriented Programming, SOA principles, and all possible Java application frameworks that have implemented the MVC design pattern.

Steve was always the only person in the building who knew exactly what the garbage collector did to the young generation. He never complained if his next client was several thousand miles away from his hometown (*Daddy's flown across the ocean leaving just a memory*).

About three years ago, by pure coincidence, Steve got a project with the same company and division where Alex has been working all these years. He was one of hundreds vice presidents with a six-figure salary, wearing an expensive suit, Six Sigma brown belt, and matching shoes. These friends were happy to work with each other, but this did not last long. The firm decided to lay off several hundred employees and let go of most of the contractors.

Alex was too expensive for the firm and Steve's contract ended sooner than expected (*All in all you were all just bricks in the wall*). Alex received a decent severance package that allowed him to spend the next six months brushing up his Java skills. Steve did not get any compensation but found a new gig pretty quickly in two months.



So what's the moral of this story?

If you're young and ambitious, spend at least some time working as a contractor. Do not be afraid to start from a clean slate every now and then; this is what capitalism is all about. Besides, the average length of full-time employment for young programmers is also not more than two to four years. As you get older, you'll experience difficulties in finding pure programmer's jobs (*Hey you! Out there in the cold getting lonely, getting old, can you feel me*).

But there are always exceptions to the rules. I’ll tell you later about a 72-year old mainframe contract programmer (God bless America). Of course, he can't write as many “if-else” statements per minute as a college graduate, but he knows his application inside out, and the firm keeps extending his contracts, and gives him the most complex assignments. I’ll tell you about this guy later in the book.

If you prefer full-time employment, be loyal to the company you work for. The firm's interests should take priority over your personal goals, but don't get lazy. Keep your technical skills up to date; read professional books and magazines; and visit Java online forums on a regular basis.

During difficult times your employer will let you go without thinking twice: this is also what capitalism is about. Gurus will have to go because their salaries are too high, and junior developers will be replaced by an inexpensive workforce overseas. But this is okay as long as you are technically sound, have a positive attitude toward life, and accept that all in all you were all just bricks in the wall. I did not say this. Pink Floyd did.

Another Brick in the Wall, Pink Floyd, The Wall

Daddy's flown across the ocean
Leaving just a memory
Snapshot in the family album
Daddy what else did you leave for me?
Daddy, what'd'ja leave behind for me?!?
All in all it was just a brick in the wall.
All in all it was all just bricks in the wall.

"You! Yes, you! Stand still laddy!"

We don't need no education
We dont need no thought control
No dark sarcasm in the classroom
Teachers leave them kids alone
Hey! Teachers! Leave them kids alone!
All in all it's just another brick in the wall.
All in all you're just another brick in the wall.

We don't need no education
We don’t need no thought control
No dark sarcasm in the classroom
Teachers leave them kids alone
Hey! Teachers! Leave them kids alone!
All in all it's just another brick in the wall.
All in all you're just another brick in the wall.

"Wrong, Do it again!"
"If you don't eat yer meat, you can't have any pudding. How can you have any pudding if you don't eat yer meat?"
"You! Yes, you behind the bikesheds, stand still laddy!"

Polyglot programmers minus SQL

In the mid nineties, IT job market was good. PowerBuilder or Visual Basic plus SQL would get you employed in no time. Good old client/server days… Lots of mainframe programmers were easily surviving knowing nothing but Cobol and SQL (DB2). Two programming languages was all you need. When multi-tier architecture became hot and J2EE came into picture, all of a sudden you'd have to learn a lot more languages and technologies, for example, Java, SQL, HTML, JavaScript, XML, JSP, EJB, JMS etc. And I've learned all this jazz.

Five years ago the IT job market was really bad. Five years ago job postings would require knowledge of ten different programming languages, and if you knew only nine, you could not get a job interview let alone job. At the time I've been working as an independent contractor, but the job market was so bad that I couldn't find a decent contract and became a full time employee of a major bank, where I spent about two years. On Fridays, I'd send out an email to everyone in our department with a little SQL puzzle. Some of them I've been inventing myself, some googled up but most of them I'd take from an excellent book by Joe Celko called "SQL for Smarties" http://www.amazon.com/Joe-Celkos-SQL-Smarties-Programming/dp/1558605762 . These emails were well received and people were responding with the answers written in SQL.

SQL was still in favor. Technical job interviews would include a couple of SQL questions. It's hard to believe, but people knew how to find duplicates in a database table by manually writing "group by" and "having" clauses.

How many people these days write any SQL statements? Why bother, an ORM framework will let me map Java class attributes to the database table columns... I'm drowning in XML now.

I never liked ORM. I trust SQL. Surprisingly, the young generation doesn't mind being polyglot programmers as long as the set does not include SQL. The popularity of this language is comparable with the popularity of Latin and Esperanto in the real world. Why? I don't get it. SQL is a very elegant and powerful language with an excellent ROI!

Note. In the next two paragraphs I'll be bragging, so you might want to skip them.

In 1997, I was hired for a PowerBuilder/SQL job by a small company that was developing software for telecommunications giants like South Bell. On my first week on the project, Sarah, the co-owner of the firm was absent – she was delivering a baby. I had to wait for her as she was supposed to give me an assignment. Someone showed me a report written as a Sybase stored procedure. This daily report would run for an hour collecting various data about activities of the field technicians. This report was poorly written - it was using several cursors that were making multiple passes through the same data set. I've eliminated most of the cursors by re-writing the "where" clause in the main SQL statement and applying some characteristic functions. The execution time of this report went down from an hour to under a minute. Everyone was impressed. I became a proven commodity and spent a year in this company enjoying an easy contract with high pay check till the company went belly up without paying me the final check. Talking about the power of SQL!

Here's one more interesting detail. When Sarah came back from her short maternity leave, someone delivered the great news to her, "Yakov modified that slow report, and now it only takes a minute to run!" She looked at me and said, "Working with SQL was not your job, but I'm not angry with you – I'm too long in this industry". A couple of days later, I found out that Sarah was the original author of that stored procedure and my bad behavior showed here little weakness. Customer's interests often have lower priority than a someone's ego, but that's another subject.

In one of my mid-nineties jobs I met a very good programmer named Roman D. who introduced me to characteristic functions in SQL (they were described in this book). These functions are not easy to grasp, but when you get it, your SQL will work a lot faster. Roman was a seasoned consultant, and he shared with me an important technique for passing technical job interviews. He'd explain characteristic functions to the interviewers, they were impressed and would extend him an offer. I said, "Roman, nobody knows about these characteristic functions, and the chances are less than slim that someone would ask you about them during the interview." He smiled to me and said, "I do not wait till someone asks me about them. It's my strong point, and I always find a way to change the subject and show these SQL tricks." I've mentioned this interviewing technique earlier.

May be one day the ORM tools will generate highly-optimized SQL, but it won't happen any time soon, that's for sure. Proponents of ORM tools would argue that their tools also allow manually write SQL statements in one of their XML configuration files. If this is the case, and if you are capable of writing SQL, why bother with ORM to begin with?

Let me tell you an old Jewish tale.

A poor man comes to the rabbi complaining that his family has only one small room, many kids, and almost no money. The rabbi says, "Take all your money, buy a goat, and keep the goat in your room. Come back in a month."
"But, rabbi, we don't have enough space even for us," the man said
"Just do what I say," the rabbi replied.

A month later the man comes back complaining that the goat smells and breaks everything.
"Sell the goat and come back in a month," the rabbi tells him.
A month later the man comes back to the rabbi with flowers.
"Thank you, rabbi! We're so happy the goat is out, now we have more room and some money!"

So if you are considering bringing the ORM-goat in, think twice. Or actually, don't – this way you may enjoy the moment of happiness when the goat will be out, and you'll return to SQL.

Why hire an IT contractor:

Isn't it obvious that having an employee in many cases is cheaper for a company? So why even hire these expensive IT contractors?

Skill set. Sometimes, it's hard to hire an employee with specific skills, especially if your company does not offer competitive salaries. So what do you have to do to meet the deadlines of this new project? You hire a contractor knowing that even though you'll pay more (comparing to an employee) for the same set of skills, it's a temporary solution that will allow you to complete the project on time.

Short time needs. Would you marry a carpenter if you need to replace your kitchen cabinets? And when your toilet starts leaking, will you divorce the carpenter and marry a plumber? You may not need a person with specific skills for a long period of time, so why bring him/her on board as an employee? With consultants, you do not need to go through this ugly divorce procedure. A separation can be done in a heartbeat.

Project managers want to sleep well at night. There are contractors and then there are CONTRACTORS!. Some of them are hired for a routine coding, while others just for being seasoned developers. Smart development managers keep seasoned and overqualified consultants on billing just in case. If something goes wrong, these contractors will fix the problem quickly. These managers realize that they could have gotten a person for the same job for a lot less money, but they bite the bullet just to sleep well at night.

A second opinion. Say, a new developer manager comes into an IT shop with established and known superstar-developers, authorities and other bullies. This manager wants to find out if the code of the current application is written efficiently, using progressive technologies and design patterns. How can this be done without alienating the team? Hire a consultant and ask him/her to perform a code review and offer some improvements, if needed. In the "worst case scenario", this consultant won't find anything wrong, but a negative result is also a result. The manager will have peace of mind. If this code review will find some pain points in the code, it'll help the manager to address them without being a “bad cop”.

Offshore contractors. A project manager may not have any other choice but to hire offshore software developers, which are often hired not just because it's cheaper, but because of his/her skills. In the former case, you'll wind up paying more by the end of the project. While in general, I am not too happy with the way outsourcing is handled, I’ve been working with some very good developers located overseas. I've interviewed them for the job, I liked them, and they did a very good job.

But the main reason why a project manager hires a contractor is to help him in achieving his/her career goals. Correspondingly, the main goal of a well trained contractor is to help the hiring manager in achieving his/her career goals. Sounds cynical, doesn’t it? But I’ve warned you that this is a book about the real-world enterprise development, didn’t I?

Will high-paid contractors become extinct?

People often ask me, "How are the rates? Are they as good as last year's?" They do not necessarily want to work as contractors, but are rather looking for a confirmation that the IT market is in a good shape.

The logic is simple: if companies are willing to pay good money to contractors, then new well budgeted projects are being created, which is good for everyone.

While anyone can go to a free clinic, A medical patient may be willing to pay to a famous doctor $500 for a five-minute consultation.

While there are free public lawyers, their famous colleagues charge about $1000 an hour, and as we've seen on multiple occasions, they keep their clients out of jail.

This applies to the software development as well. Employers could have hired five contractors overseas instead of hiring this local expensive one… Are they wasting money? I do not think so. On a slow day such contractor may be just browsing the Internet, while ten software developers in Bangalore are typing if-else statements at the speed of 20 words per minute. But guess what, once in a while he can suggest a solution that will save this employer tons of money by creating a scalable and highly available system. And experienced project managers know this.

IMHO, high paid IT contractors will never become extinct regardless of flourishing outsourcing or recessions. Their rates may go down 25-30% but eventually they come back as long as they maintain skills at the high and up to date level. Just read, write, code, attend professional seminars, learn hot technologies and … never stop.

You can say that you know a very experienced guru Joe Shmo, and he's out of job for a year. There are only three reasons for this to happen:

1. Joe’s rate is too high for this geographical area at the moment and he’s not willing to relocate.

2. The requested pay rate does not match Joe’s resume, but he’s not willing to accept this and lower the rate.

3. Joe’s skills became outdated while he was out of work.

Living with outsourcing

Initially, I named this section “Dealing with outsourcing”, but then I renamed it to “Living with outsourcing” because outsourcing is here to stay. It happened in other industries. For example sometime back I went to a toy store with my son and he found a box marked “Made in the USA”. My son was surprised and said, “Daddy, I thought all the toys were made in China”, and he was right. Most toys are made in China and this may happen to software as well so you just have to learn how to live with it.

First of all, I want to tell you that outsourcing is not as cheap as it sounds. Managers are happy to fix reports to show you that they can hire three people in India instead of hiring just you alone because their salaries are much smaller. Of course, the cost of living there is much smaller as well but not everybody is trying to talk about how this outsourcing requires additional expenses such as additional business analysts. These people from overseas may not know your business, and now we need to hire someone who will explain to them everything about the project and about the business.

In some cases your firm will be facing additional licensing fees as it needs to install the required software overseas. Additional tech support is also required. If something goes wrong overseas, what’s going to happen to your data? The Sarbanes-Oxley act requires extra protection for data if they are used overseas by offshore developers. It all adds up.

Outsourcing eliminates some jobs and create others. Mostly junior level positions are being outsourced - keep your technical skills in good shape.

If you don’t want to constantly learn these new technologies that come up every day or every month consider to switching to a different area. Maybe it worth pursuing a career of business analyst? In this case spend more time learning the business of your company. You will still be working with familiar people, applications plus managing offshore teams.

Training, training and training

How to get trained? There is tons of information on the internet. Find an online forum where people who use your language communicate. Learn about their problems and challenges. What’s hot, and what’s not? Buy at least five books a year and read them, study them, and if your firm offers some internal training you must use it.

Corporate IT training

When the moment is right, you can be ready
from a Cialis commercials

Back in the '90s, we became accustomed to receiving thick glossy brochures from various training companies. Five days of such instructor-led training would cost more than $2,000. For corporate employees this was "other people's money," and usually employees were entitled to at least one week of such training annually.

In '98, I finished my PowerBuilder career. I was working as an independent contractor and decided to switch to Java. I had learned the language by reading dozens of books (yes, we used to buy technical books in the last century). But when you switch from one language to another, the most valuable knowledge is not in the books. I needed to know how the real-world Java projects were designed and developed, so I paid $2,500 for a week of WebLogic training (the most popular Java application server at the time), and it was worth every penny. The instructor was a knowledgeable guy and this course was an eye-opener. I figured out what had to go in Java servlets and what went into EJBs, what is a Façade pattern, and what to watch for.

This training worked out well, because of my motivation: I needed to pay my bills, and when you apply for a Java position, your previous PowerBuilder experience (other than an understanding of OOP) doesn't count.

In 2001, the U.S. economy went into a long recession. When an enterprise goes through difficult times, its management lays off some people and immediately cuts the training budget. The mandatory trainings like Six Sigma or CMM will always survive, but the real stuff gets frozen. In the beginning of this millennium, those training companies that managed to survive reduced the tuition costs and their fat brochures turned into flyers. Course enrollment dropped drastically. They would even run classes for as little as three students. If the course was designed for five days, the corporate clients would ask for them to deliver it in three.

Less expensive online training came into the picture, but it proved to be boring and less effective than the classroom one. However, since the economy remained in recession for three years, many people suspended their computer education and started to whine about outsourcing.

Now, instructor-led training is back, and tuition is getting higher again. Guess what’s the most expensive training these days? Some companies that make free open source software, charge a premium for training: $3,000 for a five-day course per person. Well, they need to make money somehow, but I'm sure this won't last long; a new breed of startups that sells support of the open source tools will balance supply and demand by offering more reasonably priced training.

Who Is Teaching

When I was doing a contract training, it worked as follows: I was getting an e-mail with the title of the class, airplane tickets, an overnight package with a training manual, and a CD with code samples. Smaller training providers don't develop their own manuals, but purchase the courseware from third parties or the vendor of the software they teach. Once I had to deliver a one-day, MQ Series training. The manual was poorly written, but since I was right off a messaging project, I had lots of things to say on the subject. The students were happy and the class was saved.

But I have to admit that I’m guilty of teaching technologies of which I have only a book knowledge. Some instructors just read the manual and slides aloud. Their version of the manual may include additional comments that you don't see, so it looks as if they know more than you. Always ask about the instructor’s credentials before you are enrolling into his class.

Here’s my story of how I became an Adobe Certified Flex Instructor.

In the Summer of 2006 I was working with Flex at full speed. In addition to this, I wrote technical articles and co-authored of an advanced book on real-world programming with Flex and Java. I always enjoyed teaching programming, PowerBuilder, Java, and now Flex. And when you teach any programming language, you need a good text book or courseware. I decided to purchase Flex courseware from Adobe, complete all the labs and start teaching Flex to the masses. When I contacted Adobe, they answered that the courseware is available only for certified instructors. This sounded reasonable, and my next question was, "How do I get certified?"

Since I was an early Flex adopter, had published numerous articles, worked on the Flex book, knew people on the Flex team, had Java medals all over my chest, and was teaching programming at NYU, I was expecting the following answer from Adobe:

"Dear Yakov, we are so happy that you've decided to become a certified Flex instructor! We'll overnight you the diploma, and please start teaching Flex now."

The real answer was a little different:

"Yakov, get your butt on a plane and fly to Seattle next month. Sit through the five-day train-the-trainer Flex class, then prepare and teach a one hour training session in front of an audience using our courseware, and if we like it, you'll become Adobe Certified Flex Instructor".

Well, the wording was more polite, but the meaning was the same.
And I did exactly this – took a week off at work, purchased arline tickets, booked a hotel, and sat through a very intense class at Adobe. At the end of the class I ran a short class covering various units using the original courseware, and passed the test.

Is the instructor qualified to teach the next class you are planning to take? Do not be shy and figure it out in advance.

Finding quality training

Most of the large corporations have a list of approved training vendors and courses to choose from, and it seems that there is nothing you can do about it.

Wrong. DO not just blindly accept the course by an approved training vendors. Find the relevant conference or a seminar. Such seminars always have technical sessions on your technology of choice with first-class speakers who are practitioners, and many of them are book authors as well. These seminars usually run training over parallel tracks so you can pick the classes that match your objectives. These events are less expensive than comparable vendor training, and the quality is better (just try to avoid marketing presentations).

Pick a conference and remind your boss about all those long hours you've spent on the project. You need and deserve quality training!

There are lots and lots of passive people who just prefer to pray that things won't get worse, and they'll keep their jobs forever. They do not want to irritate their bosses by asking to send them for training.

A guy from a large corporation told me that everyone in his department was entitled to two weeks of free training a year, but most of them did not use it! He also revealed that he wouldn't listen to technical presentations on the conferences, if presenters do not have giveaways like free T-shirts.

Recently I've asked a couple of seasoned Java programmers what the acronyms AOP, and JBI mean. They didn't know. Ask your colleagues a similar question but use the acronyms that represent a leading edge technologies in your area. I bet most of them would not know them.

There are plenty of great programmers who just write code for their employer day in and day out. Raise your head and look around. See what other people with similar skills are up to. You may find a vibrant community of software developers that share your passion. Be a part of it.

Enjoy your technical conference

Next month I'm planning to be in San Francisco[1]. This is an exciting trip for me… sort of a high-end vacation.

"How dare you!", says Joe, another JavaOne attendee, "I'm planning on working there twelve hour days trying to absorb the technical wisdom of Java gurus. What vacation are you talking about?"

Hey Joe, just change your attitude. Do not spoil the party. JavaOne is a place with a super-high Java energy. Sort of a Java spa…Being in the place with the highest possible concentration of people sharing the passion for the same thing is like a medical procedure…Just being there will cure and recharge your worn-out brain…Enjoy the fluids…

Light breakfast in the morning, and then not more than two brain-massaging technical sessions. Make sure that most of your sessions are given by the best possible masseuses.

Who is teaching is more important than what is being taught.

Then, a two-hour lunch break. Eat slowly, enjoy your food, and stop drinking these soft killing sodas.

After lunch walk around the vendor area. Do not get intimidated. Like the design of this T-shirt? Just stop by the booth, introduce yourself and spend five minutes listening to the brouhaha about how product XYZ will revolutionize your life. Get the T-shirt and move to the next table.

Look at these nice little glowing pens! Aren't they something? Just give these vendors your business card and bring home a couple of pens for your kids. You may get this annoying phone call from their salesman in a month or so, but it's in a month... but your kids will start enjoying these pens next week. Daddy's back from a business trip! What did you get us?

[1] I've written this piece prior to attending one of the JavaOne conferences

Look at this lady in red - she carries a huge bag of freebies and brochures. Trust me, she’s not going to read them. In the best case scenario, she'll bring them to her office after the show. But most likely she'll leave them in the hotel room.

On a more serious note, if you are really interested in a particular technology, you’d better spend some time at the vendors’ tables. For example, if you are interested in Java messaging in general, stop by every company that offers their implementation of JMS. You’ll find some strong technical people who researched this particular technology really well. You may not get access to them this easily any time soon, especially if you live in a small town somewhere in Alabama.

After lunch, I prescribe up to two more technical sessions, and then blend in again with the crowd. In the evening, get a couple of beers or other adult beverages.

Attend a couple of keynote sessions. While the topic should be of interest to you, the speaker’s personality is the most important criterion. If s/he was able to get invited to give a keynote talk at a large conference, this deserves fifty minutes of your precious time. If the keynote speaker is not that great, you’ll be having a hard time tomorrow trying to recall what that motivational keynote speech was about.

Bright future? Does Java need closures? Why EJB still deserves a second look? And most importantly, did he cut his ponytail? Will he? I can’t trust my memory anymore, and will be taking notes and blogging from the show on a daily basis. I might get myself in trouble again, as it happened after I’ve asked a question and published my notes at a recent conference. Oh well…

The cost of attending a technical conference

There is a number of technical conferences where software developers and IT managers can learn about the industry trends and improve their technical skills. For example, the number one event for more than five million Java developers is an annual conference and expo called JavaOne. Each year ten to fifteen thousand developers gather in San Francisco for learning, networking and partying with similar species.

I did some math to calculate the minimum damage that attending JavaOne may cause to your valet. In my calculations I was assuming a registration fee of $2500 (waived for speakers). I did not take into account early bird discounts or any other coupons that could lower the fees). I did not include local transportation, parking, food and drink expenses, which for some people may substantially increase the cost of attendance. So let's see…

1. For a speaker (non-contractor) living in the Bay area: $0
2. For a speaker (contractor between the projects) living in the Bay area: $0
3. For a speaker (contractor on the project) living in the Bay area: $3000 - $5000 of lost earnings.
4. For a regular attendee (non- contractor) living in the Bay area: $2500
5. For a regular attendee (contractor between the projects) living in the Bay area: $2500
6. For a regular attendee (contractor on the project) living in the Bay area: $5500 - $7700 for lost earnings +registration fees.

For US attendees living in the East coast, add from $1000 to $1500 for air tickets and hotel to the above.

If you do not live in the USA, add an extra $500-$1000 for more expensive air tickets, and residents of some countries may have to shell out up to an additional $200 for the US entry visas. On a positive note, the total package price looks a bit better for people who live in the European Union, because US dollar is cheap comparing to the euro and each thousand dollars turns into "only" six hundred Euros.

If you'll add all these numbers, some people (or their employers) will spend at least $10,000 USD to attend JavaOne. And guess what, all these expenses won't stop at least 12,000 people (may be more) from coming to SF this year. They must be in love with this beast called Java. By the way, how much did you spend on you recent vacation with your spouse/boyfriend/girlfriend? Who do you love more: them or Java?

There is a group of people who will come to JavaOne, but will not attend even a single technical session. They will just hang out in the nearby bars and restaurants making new friends and contacts. For example, read this blog entry of Hani Suleiman describing one of his days at JavaOne 2007. Networking, networking, networking (remember, location, location, location)…

How comfy is your cubicle?

How comfy is your cubicle? Actually do you even have a cubicle? Is it decorated well enough?

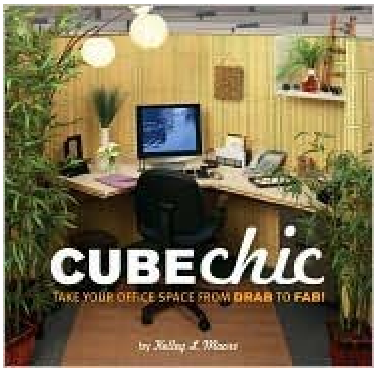
Since I’ve spent years working as a contractor, I’ve had an opportunity to work at different corporate cites. Some companies use an open space layout, where everyone can see each other, which is just horrible. In some cases you see cubes with tall divider-walls when the entire floor looks like a deserted maze.

Joel Spolsky insists that every programmer has to have a private office. This is a great idea, but the chances are close to zero that you, a programmer will have your own office. My working space during the last six months consists of two square feet area on the conference room table covered with cables, wires, and power cords. Am I complaining? Not at all. I am a consultant, which means that I need to help the client with solving THEIR problems, not mine.

An absence of working space stops me from bringing any personal belongings to work. I come with my laptop, connect to the network, put on my headphones and start working. When I leave, no cleanup is required. It’s as if I never even was there.

The other extreme on corporate floors is overly decorated cubes: tons of photos, books, about a dozen of different plants, sport memorabilia, and mugs with logos of all previous employers... When I see something like this, it seems that people think that they will live there forever. Imagine if they get fired - they'll need to bring a moving truck to take all these items home. I used the word fired on purpose - these people are not planning to leave the firm voluntarily. Ever.

Barnes and Noble bookstores offer help for people who like decorating their cells. It's called Cube Chic.



Now you can have the Garden Cube, the Cabin Cube or even the CEO Cube. I guess, your self esteem will go up if you turn your cube into a CEO-like office.

No, this is not for me. Let me push aside all these wires and power cords away, so my laptop and elbows will fit.

S/he or cherchez la femme

The Wall Street Journal has published an article called "Do women hate IT?": http://blogs.wsj.com/biztech/2007/08/08/do-women-hate-it/
Here's an extract from this article:

The percentage of women working in information-technology departments, which wasn't high to begin with, is dropping. With an IT-labor crunch looming, it's time to ask: What is it about IT that may be repelling half the population?

While women hold 51% of all professional positions in the workforce, they only made up 26% of IT pros in 2006, down from 29% in 2004, according to the National Center for Women and Information Technology. Only 13% of corporate officers at Fortune 500 tech companies are women. And Jenny Slade, communications director for the NCWIT, tells in the Business Technology Blog that women who do pursue IT careers tend to leave them at a higher rate than men.

I've attended one of the popular Sys-Con Conference called AJAXWorld. The speakers were great, but beside enjoying the presentations, I was looking for women, or as the French say, I've been cherchez-ing la femme. Just look at the picture from that conference, and see if you can spot one woman.

When I write my technical articles or blogs, I always face the dilemma of which gender to use while referring to a user. Often after the words "the user", I put "he or she will do so and so" to make sure that the readers won't assume that most of the users are males. Then I found this nice way to put it: "s/he". Some authors are afraid to be proclaimed male chauvinists so they just use "she" all the time referring to the user or a programmer. Males do no care and will never complain about such "discrimination".

During the short breaks between the AJAX seminar presentations, the video camera was showing the attendees moving around. Yes! I've spotted a couple of women! At first, I was surprised: how come all the men were dressed down, while all the women were dressed up in a tuxedo-like suites. Then I realized that these women were hotel employees serving beverages to the attendees.

Finally, I spotted two or three IT-looking women. My statistics are very subjective, but I'd say that men/women ratio at this event was about 25 to 1 and I do not see this situation changing any time soon.

Sexism, women and IT

Some women are concerned about sexism in IT. Here's what I think on this subject.

First, I do not like when people curse in public . Not just because that there are women in the audience, but it’s just plain wrong.

Second, there is very small number of women programmers. Is this because some bad guy do not let them in? I doubt it. For some reason, women decided to abandon this profession. Are there some idiots that believe that men are superior? Yes, there are, but they are minority. I’ve never heard that a woman was denied employment I n IT because of her gender.

Do women earn less money than men? Most likely. Is it because their priorities are taking care of kids and husbands/boyfriends? Most likely.
Is it wrong?

IT managers need go-getters. Most of the women need to take care of kids and us, men and there is nothing wrong with it. They can't stay late. Of course, there are exceptions. But mediocre managers do not want to take chances, and the vast majority on enterprise managers are mediocre.

I’d love to see more women in IT and wrote a section about it in this book called “S/he or cherchez la femme”.

On the other hand, I think it’s wrong that people in the USA are AFRAID to say a compliment to a woman, because they might get sued. My friend had to watch a video during her orientation day in a major corporation. They were showing several episodes. Here’s one of them.

Two men are sitting in a corp. cafeteria. A woman in a short skirt stops by a vending machine. She bends to pick whatever she purchased. The men’s heads *automatically* turn to see her yes indeed. She can files a sexual harassment suit.

Here’s another episode. A secretary makes a compliment to her boss about his neck tie. He can sue her for sexual harassment.

Is this insane or what?

I think that enterprises that force people to maintain gender-less relations are doomed.   This is not healthy and not productive. I've been working with lots of women that were great programmers. Mary, you look so good today, but please do not get me in trouble for saying this!

Arranged marriages in IT

As per Wikipedia, *"an arranged marriage is a marriage in which the marital partners are chosen by others based on considerations other than the pre-existing mutual attraction of the partners."*

This definition comes to mind when I see how large IT organizations prearrange "marriages" between the application developers and architects. I'd like to discuss potential issues between architects and developers and, to avoid confusions, I'll keep quoting Wikipedia *in italic font.*

The honeymoon

I'll be describing the situation in the Java camp, but it's applicable to any modern programming language.

As soon as your IT department grows to more than a half of a dozen Java developers, the leader of the pack (the Architect) suggests a centralized creation of reusable components. This is an easy sell: your group is agile and still not too large and, if one of these components needs to be changed, the architect is right on the premises and s/he works for you and on your schedule. Developers know on which shelf the Singleton and Data Transfer Objects reside and where the date transformation utilities are located, and they are reusing them as the need arises. At this stage we can call relations between developers and architects consensual.

The family life (a.k.a. corporate politics)

Time goes by, and older Java species bring in the younger ones. The population increases. Management moves the architects from several application development units into a new group where they can increase reusability of the objects and frameworks across the enterprise.

When a new development project begins, you (the application developer) are told to use only the objects and frameworks recommended by the architecture group. Basically, you don't have a choice.

Noble families, especially reigning families, long used arranged marriage to consolidate their strengths and to join their kingdoms. The parents, who often arrange the marriages, are trusted to make a match that is in the best interest of their children; though there are times when the choosers select a match that serves their interest and not the couple's.

Yes, your architects create new components and frameworks, but don't they have to compete with outside third-party vendors? If Jakarta Commons has a generic Pool object, why are you not allowed to use it in your project and have to use the homegrown pool instead?

Arranged marriages can also be very flexible. In one scenario, the parents introduce their son or daughter to several potential mates, while giving two the final decision, given some time.

Here's a typical conversation of a Java couple over the morning coffee:

- Darling, I need a generic Java class that would run SQL queries that are given in an XML file.
- No problem, honey. Now I'm working on a very exciting project: a global logger that will allow reading of any log file on any specified corporate workstation. But I'll definitely look into your request next month.
- But I have my deadlines... Remember, you promised that my wish would be your order...

Architecture groups often turn themselves into small kingdoms where mere Java mortals are not allowed (they might have picked up this infection after multiple unsafe relations with Oracle DBAs). Their main business is now the evaluation and purchasing of third-party tools and the introduction of new software layers between these tools and business applications. They know how to talk the talk, and the CIO rests assured that software architecture is taken care off. Meanwhile, experienced application developers start to quietly develop their own components to meet their deadlines. Their weak attempts to offer these completed useful components back to the architecture group are not always well received.

Proponents of arranged marriages claim that arranged marriages are more successful than other marriages. They hold that the spouses in an arranged marriage begin without any expectations from each other, and that as the relationship matures, a greater understanding between the two develops.

The family budget

Who pays the architects' salaries? The architects usually cut slices from the approved budgets for business application development. I am not against such deals as long as the architects don't forget whose funds help them to make their living. They can really save the firm's money by suggesting solutions leading to efficient utilization of existing server licenses, idling hardware, use of open source products, parallel computing, performing code reviews, mentoring of junior developers, delivering technical training (not the one that exists in the approved list of courses), and suggesting the best practices that are immediately applicable to business systems.

Arranged marriages operate on the notion that marriages are primarily an economic union or a means to have children.

Unfortunately, not every marriage produces children.

Divorce is not an option

It has also been said that in some cultures where divorce is forbidden or uncommon, arranged marriage would work out nicely because both husband and wife would accept the marriage, producing their best efforts to make it a success instead of breaking up at the slightest conflict.

Needless to say that application developers must also put their best foot forward and stop blaming architects when something doesn't work right. The chances are that you didn't spend enough time learning how to use these components. Maybe they're not that bad?

Oh well, it's time to take a shower, go to bed, and have relations with my spouse...oops...I meant to say it's time to go to a meeting with the enterprise architecture group.

Increase your visibility

Manage your manager

Being a good software developer is an important part of the game, but you should not ignore something called “managing your manager”. If you are good, you manager has to know about it.

For example, you may get a two-week assignment and report to you manager in time saying that you did it. The manager will thank you and will move on with his business-as-usual stuff.
But how about these three technical challenges you had to overcome while working on this assignment? You have to make sure that your manager knows about it. Keep track of these challenges and solutions and write an email to your manager explaining what you had to do to overcome these technical obstacles. You manager must know that it was not an easy assignment, but you’ve completed it nevertheless.

Any manager looks for people s/he can depend on, and s/he’ll respect you more knowing about your achievements. Do not just quietly do what you are told to. Do not be shy – manage your manager.

Have you published your book yet?

Most of us have purchased books online. But visiting a real bookstore can be a much better experience. This is how it goes. You slowly move your eyes along the bookshelf...Stop, let me open this one. No hurry. I believe in chemistry between books and readers: you either like the book or you don't. This very moment. Without even reading it.

No rush. Do you know that books are not put on the shelves randomly? Books on hot topics and books by well-established publishers like O'Reilly are always sitting on the shelves at your eye level. When some languages or tools are hot, books start their "career" movement up the store ladder, or rather up the shelf. Three-four years ago, Java books dominated, and then .NET started fighting for space. All of a sudden books on Ruby and AJAX popped up at the level of my belly. Then some other hot software suddenly moves from the floor level to the top shelf...

Each book has its history. Who are these people, the book authors? Why do they write books? Is it for the money? Fame? Are there any other practical reasons? What's the process between having a book idea and seeing your book on the shelf of a brick and mortar bookstore?

I'll tell you my story, since I had to answer all these questions for myself in the past. I've written several books some of which were self-published, while others were published by professional publishers.

First, I'll share with you my self-publishing experience. This is rather detailed section where are share with you my personal experience in publishing. But I really hope that this will encourage you and stop you from giving up on your dreams.

In the year of 2001, I was one of millions of Java programmers. But besides my 9-6 job, I've been teaching programming in different languages to different categories of people. Back then I've been helping adults switch their careers. They've been retrained into programmers from being electrical engineers, hairdressers, even a coal miner. In the US, you do not have to have any formal education to work as a computer programmer. If you are good technically and have some relevant industry experience, you are in. For example, right now one of my colleagues is a former US marine who never went to college. He's an interesting person and very good self-taught software engineer.

In the end of nineties and in the beginning of this century while teaching Java, I was preparing handouts for my classes. The students liked the handouts; they liked the way I explained the materials for the classes and the exercises. Some of them asked me, “Why don’t you publish these handouts as a book, a Java tutorial?” Initially I didn’t pay attention to these suggestions, but at some point I said to myself, “Maybe I should try”.

Java was pretty popular in 2001, and it was well covered in the press. There were more than a thousand books published on Java and Java-related technologies. This huge book pool also included dozens of Java tutorials written by well known authors.

And here I am, with my big idea of writing yet another Java tutorial. Who am I to write it? But why not to try? This is America, the land of opportunities, isn’t it? It sure is, but this does not means that everyone else in America has to work on making your dream come true.

Anyway, I decided to write Java tutorial number 159 or so and started sending emails to various publishing houses offering the outline of my future Java tutorial. And acquisition editors (people who are evaluating book proposals) were responding to my letters in a similar fashion, “Thank you very much for your proposal, but we are not planning to publish a Java tutorial at this time. But we’ll keep your proposal on file, and should the need arise in the future…”

One rejection after the other. But, as the old saying goes

Every rejection brings you closer to your goal

So I said to myself, “If they don’t want to publish me, I’ll publish the book myself with my own funds.” At that time I did not have any slightest idea of what I’m about to start and what it takes to write, publish and sell a book.

I started writing my tutorial. Talking about programming is a lot easier than writing about it. Add to this the fact that English isn’t my first language...

Writing a book is only one small part of the publishing business, which I had to learn too. I had to learn each step of this business – how to do editing, formatting, layouts, book cover, printing, promoting, selling, and shipping the books.

So I purchased a book "Dan Poynter's Self-Publishing Manual", which helped me learn some basics of this process. This book had some good ideas, some useful contact information, and it explained to me how to run such a project. In some cases this book would offer non applicable advices, but this is a common problem of many people who are trying to give you an advice.

By the time they give an advice, they have already achieved a lot, they know the right people, and what seems to be easy for them just isn't that easy for some guy who is just a programmer trying to publish and promote book.

But that book was definitely a useful read for me, and if you are about to self-publish a book, start with Dan's manuscript. At the very least it will put your thoughts on this business in some kind of a structure..

I knew what to put in my Java tutorial because of my classroom experience, and the reactions of my students.. So I created a book outline and a table of contents. This book would consist of two parts: the basics of the Java language and J2EE stuff (Java Servlets, JSPs, EJBs, and Java messaging). I am a strong believer that all examples in any programming class have to be from the real world. I don't like explaining business programming using examples of a bird's life, trees or apples and oranges. I prefer to giving the examples from the field of the business where my students will be working in the future.

In the first part I was planning to give lots of different examples; and in the second part readers were supposed to build an application from a financial industry, which is a useful knowledge for programmers who are looking for a job in New York City and many other financial centers in the world.

I started writing my book. The first question was what software to use to write a book. I didn't want to pay any money for expensive publishing software so I used Microsoft Word as an editor. This is not the best solution when it comes to laying out a book. But it's not publishing software, and I can't blame it for not doing something it was not created for.

While writing this book I've learned that there is a huge difference between explaining or just teaching a class and writing about the same material in a book. If you know what to say, that's one part of the deal but, when you start writing it and then re-reading it, it's a completely different story. All these little mistakes and omissions that would be forgiven when you talk pops right at you when written in the book.

I'm sure I make mistakes while speaking English but I am not the only one. Even native English speakers suffer from this. But, in writing people don't forgive mistakes that easily, and many years after the book was published, one of the buyers on Amazon gave me two stars out of five saying that the book content was great but the editing was poor. So it's very important to hire a professional proof reader, which I did not do.

I didn't plan to spend too much money; I didn't have too much money for this project. I did some evaluations of how much the book would cost and what items should be included in my expense list.

I decided to save money on proofreading and figured that my kids could do this. English was their first language. This was a mistake number one.

Then besides this expense I had to find a printer to print so many copies of the book. How many copies? I had no idea. I realized that the more copies you print, the cheaper the cost of each copy is. I decided that two thousand copies would be a reasonable number to lower the printing cost and keep me in business.

The next step was diagrams. I had to create all these diagrams, which were not an expense as I drew them myself.

The next part was the book cover. I had to think of what I was going to put on the book cover and again I didn't want to hire any professional cover designer. Cheap bastard!

My older son was in high school, and he was a good artist even back then. Later, he became a professional artist/animator. I asked for him to draw the cover. Now I understand that the cover could have been better, but hey, I saved money once again!

Then I had to decide about the size of the book. Is it going to be seven by nine inches or eight by eleven or maybe five by seven? Typically computer books are seven by nine or around this size, and the size of the book cover (front, back and the spine) should be able to hold three hundred of pages of book.

By the way, what do you put on the back cover of the book? I didn't come up with anything smarter than a picture of my smiley face with some information about myself, the book contents and a bar code. Oops…where to order the quality image of the bar code?

The book by Dan Poynter suggested that it would be nicer to put some quotes from well known people in the industry but I didn't know any well known people in the industry. How could I even approach anyone asking for a quote about my book? That was out of the questions.

To order the barcode I had to apply for an International Standard Book Number (ISBN). I found a place to apply for the ISBN numbers and they don't sell you one ISBN number they sell you chunks of at least ten numbers a piece for about $250. Why on earth would I need ten ISBN numbers? This was the only book I was planning to publish! I just didn't have a choice and bought these ten numbers.

ISBN is a unique identification number that every book should have if you are planning to sell them through the book stores in the USA. I got that part done and found a firm that knew how to create a barcode image based on ISBN. This was a small expense - ten dollars.

Now I had to find a printing company. After some preliminary online research, I found the least expensive company for printing the book, the cover and for shipping the books to my house.
Remember, I was expecting to receive two thousand copies of the book. Do you even have space in your house to store this number of books for years?

This was not a Harry Potter book and for a self-publisher, it take years to sell all the copies. I lived in a house with a basement so the storage was not an issue.

These days there are so called print-on-demand companies that can print smaller quantities of your book on an as-needed basis. It's a more expensive per-copy solution, but can work for people who can't invest several thousand dollars into a risky enterprise.

Finally, I got these books in the flesh. This was a very memorable day in my life. I did it! They said that it couldn't be done but I did it anyway! I stacked the books in the basement…and now what? Did anyone want to sell them? Not really.
It's time for the second round of rejections. I started approaching possible sellers for my book. I was sending out free copies of the book to multiple small and large stores around the country. The cover letter read something like this: "Dear bookstore owner, here is a copy for you to review and if you are interested I am willing to send more copies."

I already knew that there was a list price of the book and an actual price that you will get from the book stores or from the wholesalers, and I knew that these guys would take roughly 55% off the list price of the book. And this is what I was offering to all these small stores on campuses assuming that many colleges are studying Java and students might be interested in buying this book. This was another mistake.

Students buy books recommended by their professors. Book stores on campuses order books recommended by college professors. Other small book stores just work with a limited numbers of publishers and order books from them. They do not want to deal with small publishers.

Eventually, I figured out that all my review copies were simply sold by these book stores and they did not bother to contact me again. That was a stupid idea to send review copies to small stores because in the USA all business distribution channels are well organized, and it would take huge efforts to make them change their processes.

I needed to figure out a way to get my book to the big guys. My first target was Amazon.com, the largest super mega book store online. After browsing their Web site, I found out that they have a small publisher division. I followed the procedure, filled out some forms and sent them a review copy of my book. In two-three months they responded positively – they ALLOWED me to send a couple of copies of my book so it could be sold on Amazon.com. They also pay the publishers (at least the small ones) 55% of the list price of the book so when you see all these big discounts like 30% off, 35% off, 40% off on Amazon you shouldn't worry about them too much because they still make profit.

But Amazon doesn't want to store too many copies of your book, and they let me send just two copies. Hey, I was in a good mood – having 1998 books in my basement is better than 2000!

I sent them these books, and the book started selling… slowly. At the same time I approached Barnes and Noble. Same story, Barnes and Noble also had a small publishers division. I read about the procedure, wrote them a letter and sent them a review copy. Their process took three months or so, and they responded positively. I received a letter from them, that they were willing to offer it for sale in their store. They wanted to purchase 30 copies of my book. This was one of the happiest days in my life.

I was about to start packing the book but, unfortunately that letter had another page explaining that they do not purchase books from small publishers. There was a list of the nationwide wholesalers that B&N dealt with. Now I had to have to find one of the wholesalers willing to carry my book. Piece of cake – I already have a respectful book store confirming that they were ready to purchase 30 copies.

I started writing letters to these wholesalers. Dear wholesaler, here I am, and here's a commitment from Barnes and Noble, and would you be kind enough to be my middle man between myself and B&N. Guess what, I started receiving another round of rejection letters from wholesalers stating that I was too small of a publisher even for them! Finally I've got one positive response and Barnes and Noble started selling my books.

After selling the first thirty books they started ordering more and more. I started visiting my nearby B&N store enjoying the view of my book standing on the shelf among the big guys like O'Reilly or Wiley.

But even if your book is sitting on the store's bookshelf doesn't mean that people will start buying it unless they know the publisher or the author. In my case, they did not know either. I decided to invest in self-promotion and become more visible by approaching technical magazines. I started writing technical articles that would include my BIO, which mentioned that I was the author of the "Java Tutorial for the Real World".

I submitted abstracts of various articles to Java-related publications, and Java Developer's Journal (Sys-Con Media) responded positively. It was an article about the difference of using abstract classes and interfaces in Java - a technical area that not a lot of people understand too well. JDJ was the most readable Java publication in the world, so I considered myself lucky! After waiting for several months I got back an email from the editor in chief of the magazine, Allan Williamson. He apologized for not responding earlier, and asked me to send the article in. Allan made my day!

In a month, my first article was published in print and it was well received by Java developers. Well received means that thousands of people have read it and no one said that I am an idiot. Trust me, this is a very good result in today's world, where it's so easy to leave a negative feedback online.

I wrote a second article, and then got an invite to become one of the technical editors of JDJ, and I accepted the invite. Technical editors are people who are experts in a particular area and are willing to read other peoples submissions to see if they have technical merits. Technical editors also say yey or nay. After approving an abstract of an article, I would start working with the author on the first draft. I'd read and commented the article and sent it back to the author, then another cycle, and eventually I'd submit it for publication.

I also wrote my own articles, which definitely helped in selling my first book.

The fact that I've authored a book, allowed me to join a mailing list of computer book authors. One day, James McGovern, a published author posted an invitation to Java authors to join the team working on the J2EE Bible to be published by Wiley Publishing House. He did not say that self-published authors could not apply. I sent a copy of my Java tutorial to James and he let me in!

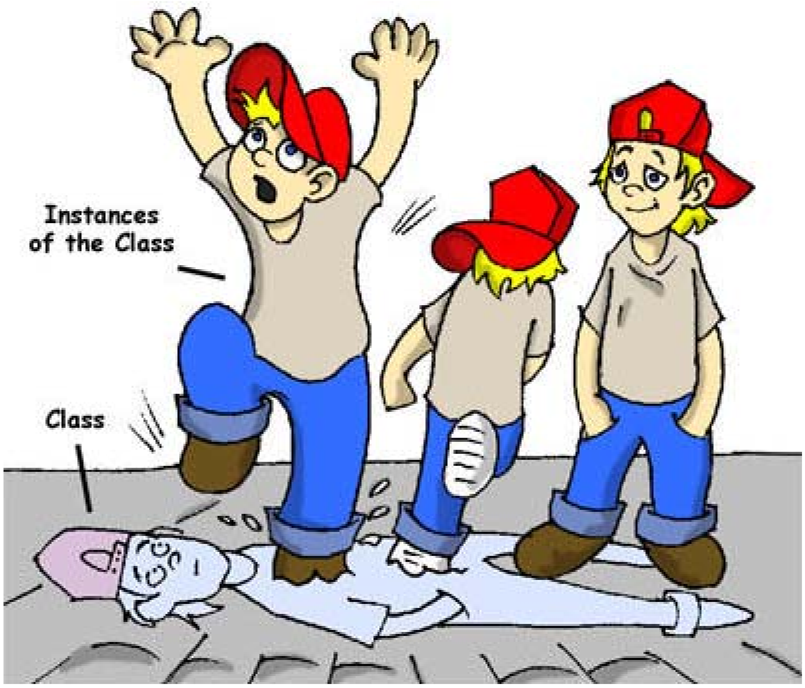
I wrote five chapters of the J2EE Bible book. This was a completely different experience and writing a book for a well known publisher is a lot easier. First, they gave me some money upfront -an advance. But I'm not going to snow you here, writing technical books isn't a profitable business. It won't make you rich. When you write a book, you literally work for about a dollar an hour. You spend most of your evenings and wee hours writing this book. Your family is not happy about that. When you read in an acknowledgement section, "Thanks to my family for support", this actually means "Thanks for not divorcing me".

Working for a professional publisher was a totally different experience. We had proofread editors and requisition editors. I did not have to worry about the book cover, the selling price, storing the books in my basement. The publisher took care of all this. But marketing… not really. When you are being published by a book publisher don't expect that when your book is printed they'd put some extra effort into marketing YOUR book.

So that book was my second book, then it was 2003, and my younger son who was about ten years old came to my office saying, "Dad, I want to learn how to program games. Can you teach me?" I surfed the Internet trying to find a book on programming for kids, and to my surprise there was nothing available for this audience. Kids who are under five years old would be reading Doctor Sues kinds of books. "Hey little buddy, this is a computer, this is a monitor, this is a mouse and a keyboard".

A whole generation of young people who are ten to eighteen years old was not considered as a market for books on computer programming. It's assumed that when the time comes, these kids will become college students and read adult books on programming. When I figured out that there was nothing to purchase for my son, I decided to write such a book myself.

But a book on Java programming for kids was supposed to be written in a different manner - an easy reading (yeah, right) with good illustrations. My older son Yuri was getting better and better in drawing, and I asked him to come up with a cartoon-like characters as original illustrations for the book. They should be funny and colorful.



He did the drawings and came up with the characters and they came out well. I wrote that book, and it was originally called "The Java Tutorial for Kids", but then I've added "…Parents and Grandparents" because when the book was written in a very simple language and could be useful for many people who are new to programming. Eventually, many adults sent me "thank you" email confirming that it was an easy reading if that is at all possible when it comes to programming.

The book (PDF) was ready, and so was I for the next round of rejections. This time I rejected to print it in black and white (this book was for kids, remember?) The publishers rejected printing it in color (too expensive), plus it's a new market for them.

This time I decided to sell it as an e-Book in PDF format, and I did it for three years. Now it's available for free download at www.faratasystems.com.

This web site is belongs to our company, Farata Systems, and I co-authored yet another book with two great programmers Victor Rasputnis and Anatole Tartakovsky. This book was about programming with Adobe Flex and Java. We rightly predicted that this technology would be hot. Our thanks to Sys-Con Media that quickly agreed to publish this book ("Rich Internet Applications with Adobe Flex and Java").

We spent a year on writing this book, which was the most difficult process so far. This was an advanced book and we are very pleased with the outcome.

The choice of publisher was also right. Even though it is advised to publish books only with specialized book publishers, we've chosen Sys-Con Media, which is mainly publishes technical magazines and organizes technical conferences (see www.sys-con.com). In terms of promoting the book, they had huge advantage comparing with traditional publisher. They've published excerpts from the book on their Web site, sent out newsletter announcing the book, and let us promote the book at their conferences. As the result, this book has purchased by software developers in more than 60 countries.

Every time I finish a book I say to myself, "Enough, this is my last book", which is a lie. This year I've rejected a couple of offers to write another technical book, and wrote this one instead.

Why did you have to read this rant about Yakov, the great book author? Because I want to encourage you to write your book too. It is not easy but doable, and after completing such a project you'll bring yourself to a different level. In most cases, being a published author or co-author looks good on your resume. Be prepared though, that in some cases you may have hard times during technical job interviews, because some interviewers will give you a special attitude, "OK, you wrote a book, let me see if you know everything…"

Luckily, most of the people are not like that. They respect you for the hard work and long hours you spent researching, writing and publishing a book.

Write your book. If nobody wants to publish it, publish it yourself. Can't publish? Create a Web site. The chances are that your book may change your professional life for the better. Go for it.

Outsourcing

In the perfect world, you can find local resources for your project. But in the USA selecting programming as a profession is not as appealing as it used to be 10 years ago, and you may have to hire an offshore team.

When a budget of the next project is approved, Frank, the manager has to staff it. The budget is small, the leftovers of American programmers are either expensive or just not available. Frank starts looking elsewhere. The largest pool of people who can write if-statements is in India. Of course it's better to have programmers in flesh in the office, but…no money, no honey. The major part of software development in Frank's project was outsourced.

The world is round

In a typical US enterprise Jimmy's manager does not fly to India to pick developers for the next project. He's given the name of one or two consulting companies to choose from, and hopefully, a chance to interview future team members over the phone.

Sure enough, someone from your firm visited India to establish the partnership with one of the major body shops there, which similarly to American ones are made up of 20% good and 80% poor to mediocre programmers. Sure enough, India's IT giants impressed your firm's executive by putting together a well done PowerPoint presentation, showing well equipped offices in a hi-tech building in Bangalore, and applied some other techniques.

Business Times writes about donkeys that complicate morning commute to the work of programmers in India (see http://www.businessweek.com/magazine/content/07_12/b4026001.htm):

" *When foreigners say Bangalore is India's version of Silicon Valley, the high-tech office park called Electronics City is what they're often thinking of. But however much Californians might hate traffic-clogged Route 101, the main drag though the Valley, it has nothing on Hosur Road. This potholed, four-*

lane stretch of gritty pavement—the primary access to Electronics City—is pure chaos. Cars, trucks, buses, motorcycles, taxis, rickshaws, cows, donkeys, and dogs jostle for every inch of the roadway as horns blare and brakes squeal. Drivers run red lights and jam their vehicles into any available space, paying no mind to pedestrians clustered desperately on median strips like shipwrecked sailors."

Thomas Friedman, the journalist from New York Times the author of a bestseller "The world is flat" was very impressed by visiting Infosys, a large offshore firm in India.…

Thomas Friedman has seen all this, but he was very impressed by visiting Infosys offices in Bangalore. They showed him "*a global conference center – ground zero of the Indian outsourcing industry' It was a cavernous wood-paneled room that looked like a tiered classroom from an Ivy-League law school. On one end was a massive wall-size screen and overhead there were cameras in the ceiling for teleconferencing*".
Now Mr. Friedman believes that it does not matter where you are, you can work on a global project without leaving your country.

His host named Nilekani, said, "*So this is our conference room, probably the largest screen in Asia – this is forty digital screens [put together] " Nicekani proudly explained, pointing to the biggest flat screen TV I had ever seen. Infosys, he said can hold a virtual meeting with the key players from its entire global chain for any project at any time on that supersize screen*".

This gets Mr. Friedman excited. I'm wondering though, how a large TV screen can impress a New Yorker? Has he ever been to Times Square located within a two minute walk from the New York Times building?

Several times Thomas exclaims in his book, "The world is flat!", meaning that these days there is no difference which part of the world you are getting services from. No, Thomas, "The world is still round!"

A Large screen TV is the last thing that comes to mind of Jimmy-the-programmer, who has to deal with miscommunication and misunderstanding of functional specs and coding assignments and delays caused by time and cultural differences.

In this section I'll try to explain why outsourcing does not work and why at the same time it's unavoidable. I fully understand that it takes two to tango, and if something does not work out between offshore and US teams, it's the fault of both parties.

In May of 2008, I had a chance to visit India – I was invited to speak at a conference in Bangalore. One day I'll include a new section in this book summarizing my IT discoveries, but meanwhile you can read my raw travel notes in this blog.

I know lots of good programmers from India, Russia and China, but the purpose of the following pages is to show that outsourcing is not as rosy and fluffy as journalists-dilettantes often present. Let's see how it works in a real enterprise IT.

What CIO should know about outsourcing

Jimmy's manager Frank started the meeting by saying that the budget for the new project had been approved, but half of the project will be outsourced to their long time partner from overseas. Can you imagine, their rates for Java programmers can go as low as $20 an hour! We can hire a team leader for $50K a year!

"No, we're not losing anyone from our team, and you should take it as an opportunity to work as team leaders, helping our new partners to hit the ground running. No, this wasn't my decision; it came from above".

Three Months Later

Mary, "*I've asked them to add two fields to a JTable on the Invoice screen. The data is being retrieved from our database so they'd need to modify an SQL query as well. I've sent them this e-mail yesterday, but it was night time over there, so they've responded today asking me to send them the modified SQL and write the name of the Java class and method where this new code should reside. I could've done this by myself in two hours*".

Frank, "*Just be patient, it's a new application for them. By the way, I'd appreciate it if you could stay a little longer today. We're having a meeting with our colleagues from overseas, but there's a time difference, you know… No worries, they're willing to come to work early, so we're starting our meeting at 7pm*".

Six Months Later

Frank, "*The system has to go to UAT in two weeks. We've all worked hard; our remote colleagues put in lots of overtime. John, you're our Java expert, and you've spent the last two weeks doing the code review of that module. Why does it work a little slow?*"

Jimmy, "*Well, that module isn't written in Java. I mean, they were using Java syntax, but it wasn't Java programming. There are chunks of unused code fragments, the code isn't object-oriented, they used the wrong Java collections, and there's unnecessary synchronization all over the place. But I can re-write the entire piece in three weeks*".

Frank, "*OK, let's do it - but quietly*".

After spending many nights in the office, the project was saved. Frank got promoted for delivering the project almost on time and showing "strong leadership in managing cost-saving external resources." But the team's morale went down the drain; two local resources (a. k. a. Jimmy and Mary) got small bonuses and started looking for new jobs.

Post-Mortem Analysis

Unfortunately, more and more CIOs believe that computer programming is a commodity skill that can be bought cheaply when needed and replaced easily like a receptionist, mailman, or any other clerk. They don't believe that having a pool of knowledgeable and talented developers adds any value to the organization.

This wouldn't be the case if the development managers (the Franks) explained to them the price that's paid for the success of such projects. But most of these managers never do this, because of conflict of interest: Frank's only goal is his smooth movement up the corporate ladder. Moreover, to increase his importance, Frank inflates the resources needed for the project on purpose.

The CIO doesn't have the budget for several additional $70K-a-year developers, so he settles on the same number of $30K developers from overseas with similar résumés. Realistically, the "cheap" labor is actually an additional expense on top of the salaries of local employees.

Another hidden expense is the extra time spent writing super-detailed functional specifications and validating the overseas work. Here's one more: for security reasons, you may have to create and maintain a separate encrypted version of your database for the offshore team.

Having said all this, I can't blame the overseas developers. Their countries are experiencing a golden IT rush, so young kids are ready to dive into muddy Java waters after spending several months in vocational school. (If I were in their shoes I'd do the same thing).

They put in long hours trying to learn programming and the business of their rich clients (not to be confused with "fat clients") on the run. As a result of this IT boom, the turnover rate in offshore teams can be as high as 100%. You can often see it just by looking at the source code. Sometimes you get a feeling that a 200-line Java program was written by 10 different people of different qualifications. Forget about naming conventions, design patterns, or any programming style.

Hey, Frank, if you need seven people for a project, have the guts to say seven and not 10. Yes, you won't have a chance to manage an international (or as they like to say global) project, but you'll definitely sleep better at night. Before giving a chunk of your project to a company overseas, talk to your developers and ask them if they really need this help. Your developers are human beings and not just nameless resources.

On the other hand, outsourcing works fine for small businesses because both parties know that the owners of such businesses count their money and won't pay for poor-quality jobs. It also works when you hire an offshore team of senior people who know the business you're in.

No, their rates aren't cheap, and don't have to be! But such teams usually consist of professionals, who take pride in their work, deliver on time without putting an extra burden on your own developers, and even mentor your staff. This is the outsourcing I vote for, but I'm not the CIO of your company.

Ten tips on dealing with offshore developers

Below is a list of tips for a rookie development manager that has to work with offshore software developers.

1. If your outsourcing partner offers you a pre-staffed offshore team, most likely you are screwed. A typical team in any country (USA included) operates under 80/20 rule – only 20% are delivering. Do not accept a team just because it's ready. Build the team yourself.

2. Put every candidate through at least the same vigorous interview process as you practice with your local candidates. You'll be surprised, but some US firms would go easier on the offshore team members just because the team was given to them from above.

3. Do not leave your office until you know what the offshore team will be working on tomorrow.

4. Do not let the local geek manage an offshore team – geeks are interested only in cool techniques and coding. They do not really care that spending a week on finding the most elegant solution jeopardizes the project deadlines.

5. Cut the losses quickly – if you hired a new offshore member and he did not deliver within the first two weeks, cut him loose. Cutting the umbilical cord sooner saves money (actually a lot of money).

6. Unless you really know the remote developers, use offshore teams for fixed-price projects only. This will allow you to better predict the final cost of the project, and if the project will not be delivered to your specification, you do not have to pay for the poor job. So called “Time and Material” projects pay per hour or per day. This is a good option if you just want to keep a particular individual around because you know that s/he has good skills, (for example, production support, multiple small projects).

Fixed-price projects require more work from your side during the initiation of the project – you’ll have to actually take the time and think what you are planning to develop, create a functional specification that reflects the final product. If you made a mistake and want to change your spec down the road, the offshore team can hold you liable and ask for a budget increase or scope reduction.

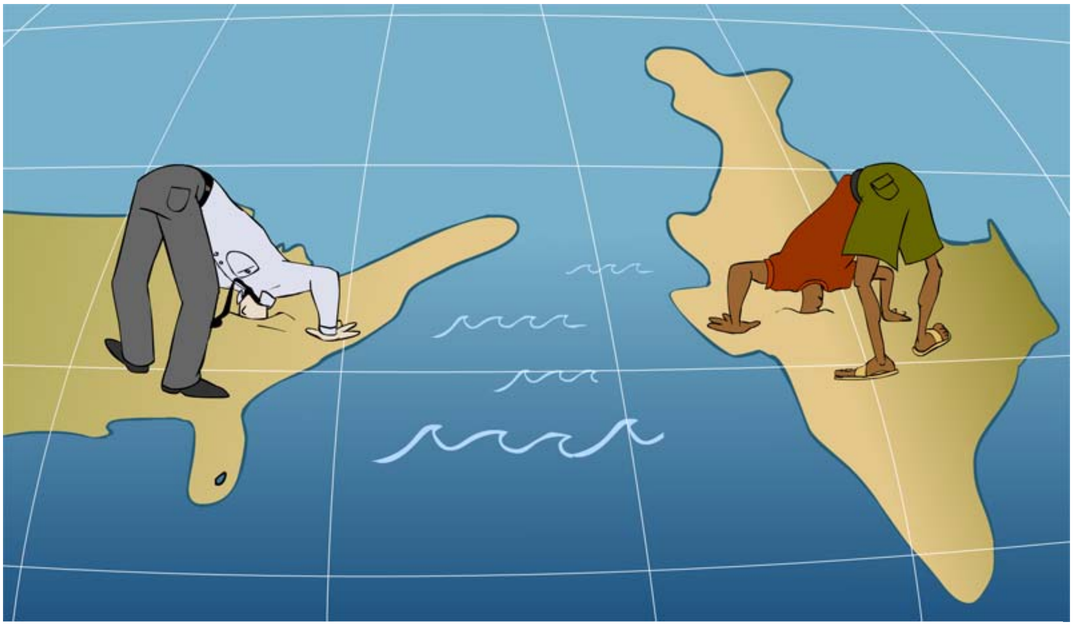
7. Do not use the same team on the new project just because you’ve worked with these people in the past. Their technical skills may not match your new requirements. Remember, the offshore team is a consulting firm, they are not your employees and you do not have to use them for all new projects.

8. When you give an assignment, make sure that the other side understands exactly what has to be done and by when. You may get the “Yes, Sir” answer, but when the delivery time comes, you’ll realize the other party did not even understand what had to be done. The “Yes, Sir” answer is typical for the Indian culture, where people are used to respect --parents, older people and higher-ups. They tend to avoiding saying “No”.

Another scenario is when you give an assignment, the other party decides to do “more” than you asked for, which leads to increased project scope and missed deadlines. This is what you may expect while working with programmers from the former Soviet Union. Many of them are still writing their own compilers on a regular basis. Time management is an issue there.

9. Make a habit to have a quick 10-min morning conference call with that remote team of telecommuters. (You, on the other hand, can just dream of working remotely. Find out what are their issues. Do not postpone these meeting to the end of the week – too much time will be lost.

10. If your outsourced project failed, and you are trying to find who is to blame for it, look in the mirror. Do not blame the offshore team.
Hiring an offshore team is just the beginning and requires extra hard work on your side. Do not just hide your head in the sand – the problems will only worsen.



Dead souls from overseas

Here's another caveat of managing offshore teams. To make this discussion a bit more interesting, let's go back in time into the first half of the 19th century.

The novel "Dead Souls" by the Russian writer Nikolai Gogol was published in 1842. At that time, landowners paid taxes based on the number of people registered with their properties. Often, landowners were taxed even for the dead souls after people would pass away.

Guess what? One entrepreneur named Chichikov started visiting landowners offering to purchase dead souls from them, as this would lower their taxes. Why Chichikov would need to have legal rights to these dead souls? C'mon, it's elementary, Watson! He wanted to inflate his importance in the society by showing the large number of souls he owned, so he could take a large loan against them.

By a magic wand, we are back in the 21st century. You are a team leader (a technical guy) on a project that includes both local programmers and an offshore team of Java developers. The boss said that the management has decided to hire a large and well known offshore corporation called Shmata Consulting. Your firm can hire three Shmata's programmers for the salary of one American with the same skills.

Things may not be as rosy as they look on paper, but here's yet another twist to it. While technical leaders work harder as they need to find a substantial chunk of additional time EACH DAY explaining to the offshore team how to write code and fix errors, the managers of Shmata Consulting do not forget to submit their time shits (did I spell it wrong?) to Frank, the manager for approval. By now, you should be able to guess who these dead souls may be.

Jimmy believes that three programmers from overseas are coding for your project, but Frank might be receiving (and signing) timesheets (yes, this is the correct spelling) of 5 people, and each of these souls were working their dead asses off putting in 80 hours a week. Talk about cheap labor. Talk about Chichikov of the 21st Century. Two centuries ago Mr. Chichikov has founded the Dead Souls Movement, without even realizing how to do these kind of things on a large scale. There was no Internet in Russia circa 1842, so his "crimes" sound like an innocent joke of a kindergarten boy.

Anyway, what's the bottom line? Is there anything you (the tech lead) can do about it? Yes just follow the ten advises from the previous section. These simple rules should prevent your project from being yet another failure with "cheaper" (but still inflated) cost. You may not like this kind of job, but at least you'll know that you are in better control of live souls, and if some little dead soul sneaks in, kill it again. You can't get convicted for a murder of someone who was already dead.

Outsourcing to students

I'm teaching software programming part time at New York University. While this and other universities offer graduate-level programs with many interesting courses, the situation is different in the undergrad world.

Some schools keep teaching how to multiply matrices in Ada or work with algebraic expressions in Prolog. Half of the courses are preparing professionals who will be operating on another planet. Information systems programs look a little more down to earth. Meanwhile, many college graduates are having a hard time finding their first jobs because many entry-level programmers jobs are being outsourced overseas, and it'll stay this way as long as it makes financial sense for businesses. Unfortunately, student loans have not been outsourced...

Remember that Catch 22 that so many recent graduates face upon facing the job market – can't get a job without experience; can't get experience without the job?

I have a plan: instead of outsourcing projects to developing countries, businesses should offer them to the local colleges. I'm not talking about simple pilot or proof-of-concept projects; I mean the real ones. This plan requires commitment and the cooperation of academia and businesses. These are some thoughts that come to mind:

- ➢ Colleges have to include more classes on software engineering and modern technologies in their undergraduate programs. Here are some of the candidates: Application Servers, Service-Oriented Architecture, Applying Design Patterns, Data Modeling, Business Intelligence, Web 2.0.

- ➢ Colleges form teams of programmers starting from the students' junior year. Faculty members lead these teams. Information about these teams (résumés, previous projects, GPAs) has to be published on the Internet and be publicly available, and businesses need to publish their project descriptions so student teams can bid on these projects.

- ➢ Colleges make their labs, networks, and support personnel available for the teams. If needed, businesses can lease additional hardware to the college for the duration of the project.

- ➢ Most of the students study Java programming during their freshman and sophomore years. Many Java components are available for free or through open source licenses: IDE, version control systems, project build tools, bug reporting systems, application servers, etc.

Businesses will purchase any additional required software for a fraction of the cost using heavily discounted academic prices.

- Business managers pick and interview teams for their projects based on the college reputation, available skill sets, location, and other criteria.

- Corporate lawyers prepare a contract with a selected team that defines the obligations of each party, deliverables, cost of development, and potential penalties.

- The turnover rate is usually high on the projects that are outsourced to developing countries, which won't be the case with student teams. On the other hand, there is a risk of not having developers during midterms and final exams. However, since the exam schedules are known in advance, the "freeze time" can be planned accordingly.

- Most of the business managers dealing with developers from other countries complain that cultural differences are a huge problem. Guess what? This won't be a problem if you outsource the project to students who live in the same country and speak your language.

- Even though students will get a minimum salary for this work, they should also earn academic credits and get graded while working on such projects.

The funny (or sad) part is that the students themselves are already outsourcing their college assignments. There are Web sites where you can hire a coder for any assignment in various programming languages. No job is too small. People from around the world can bid on these projects, and since the offered prices go as low as $10 USD per hour. It's clear that only programmers from the developing countries (India, Russia, China, et al) would be interested in these jobs.

Academic outsourcing may be even more damaging than industrial outsourcing, because rich students can improve their grades and earn their degrees without having a good knowledge of their required subjects.

Spending more time working as teams in the labs under the supervision of a faculty member or business manager will help minimize academic cheating.

There is one more secret key to the success of commercial projects developed by students: pizza! Each day the client company can send a couple of pies (half plain and half pepperoni) to the labs where the students work. They are going to work for food...and experience. It's a win-win situation for everybody.

Soviet Programmers

The word "outsourcing" is being associated with India. No wonder -- these guys are street smart, they were raised in a democratic country, they are mobile, know what’s hot on the market, they speak English and are ready to start working tomorrow anywhere in the world sharing an apartment with five other programmers.

In the USA, all people who speak the Russian language are called Russians regardless if they are Russian, Ukrainian, Jewish, or of any other descent. On the same note, if you are a US citizen, you are American regardless of where your ancestors from. We’ll follow the same convention, even though the proper way to categorize them would by Soviet programmers for all who came to the USA before 1991 and Russian, Ukrainian et al. thereafter.

I used to be one of the Soviet programmers in the 80s. After the collapse of the Soviet Union, society started its painful transition to capitalism. During more than 70 years of dictatorship, people learned the rules of the game, they knew when to keep their mouth shut, how to put scarcely available food on the table, but they lost the spirit of entrepreneurship, which started to change when the Soviet Union ceased to exist.

While Western people were differently-rich, Soviet people were equally-poor. They had pretty much the same income, small but free apartments, good education and OK medicine. Professional emigration from the former USSR was minimal.

All this started to change in early 90th. Now computer professionals who had enough money and wanted to work abroad could do it. While Russians are considered one of the top coders, they are lagging behind Indians in the outsourcing arena. I see several reasons for this:

1. For years, English is the primary language in Indian colleges, while Russians were under impression that sooner or later the rest of the world would learn the Russian language. Also, in the USSR the language of a "potential enemy" was German, so English as a second language was not too popular. This is changing now, but it takes time.

2. People in India do not think that they are the best nation in the world, while many Russians still do.

3. Many years of dictatorship have encoded the following rule in the minds of most of the Russian population: "There are only two opinions: right and wrong". This was applied to any aspect of lives including computer programming. I remember teaching programming classes for Russian immigrants, and the faces of some of the newly arrived programmers were turning purple should I've said something they did not agree with.

4. Some Russians programmers do not trust the programs written by others. They are ready to reinvent the wheel on a daily basis. For example, some of them would rather create their own libraries of reusable objects than use someone else's (see item 12 from the humorous post about Russian programmers at http://www.softpanorama.org/Bulletin/Humor/russian_programmers.shtml).

5. On the other hand, I had and still have the honor of working with extremely bright software engineers from Russia, whose skills are top notch.

Slowly, but surely the situation is changing. Programmers living in Moscow and St. Petersburg earn a lot more than their colleagues in the rest of the country. I do not have official numbers, but based on online posts on Russian Web sites, the annual salary of a mid-level Java programmer is $20-$25K. Senior programmers make $40-$50K. But the skill set of a $50K programmer living in Russia, is comparable to the skill set of a $100K US programmer (and, adjusting for the cost of living, so are their salaries). Many programmers living in large Russian cities can read English. Speaking skills are improving too.

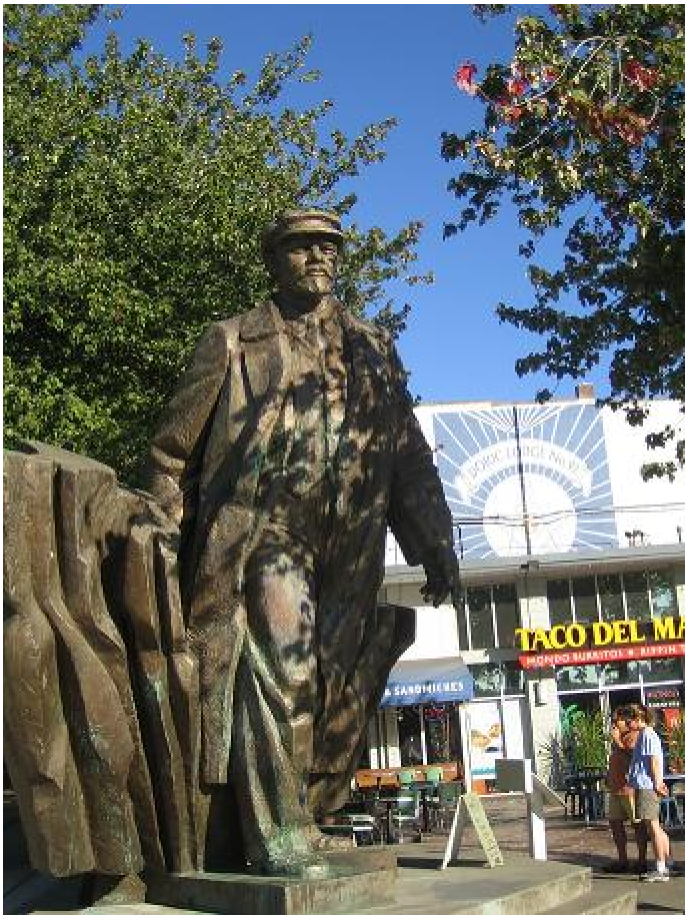
But the Russian outsourcing model has to be a little different comparing to the Indian one. You can get the best value by dealing with high-end software developers. I know the CEO of one of the most successful Russian IT companies that builds its business not by selling cheap low-level programming to the West, but by focusing on specific business verticals.

They've established their own tuition in one of the best universities to attract and "breed" the most talented people. You can't hire three of their employees for a salary of one American programmer, but these people are just brilliant.

The United States of America is still one of the leading world economies with high living standards, and it remains a desired destination for people who are not afraid of the challenges of living in a foreign land. On the other hand, the majority of Russian programmers would not leave their country, enjoying modest living standards in familiar environment.

The world's leading technology firms feel themselves at home in Russia. IBM is pretty happy there , one of the insiders from Sun Microsystems told me that they consider Russia as one their most important markets (Brazil is also on Sun's radar).

To finish with the Soviets, I'll tell you an one more story - I saw Lenin in September of 2006 in Seattle, WA, USA. Seeing this huge Lenin monument casually standing on one of the streets of Seattle, was a real surprise for me to say the least. People who are as old as Internet may not know who Lenin was, so this is especially for you: would you be surprised to see a 7-ton monument of Fidel Castro standing on the street of New York? Just take a look (this is Vladimir Lenin):



Here's the story. This 16-ft statue was erected (if you do not know the word, think of Viagra) in Slovakia back in 1988, and in 1989 it was toppled and was just laying face down on the ground. An American entrepreneur Lewis Carpenter ran into it and like it a lot. I mean really liked it, so he even mortgaged his house to buy and transport the statue to Seattle. This statue is a bit unusual, because it shows the leader or the world proletariat not just holding a book or surrounded by smiling pioneers (a communist-crafted boy/girl scout organization), but standing in flames and guns. Anyway, for 28 Grand Lewis brought it to the USA. Unfortunately he died in a car accident in 1994 leaving his wife with an unpaid mortgage and Lenin.

Vladimir Lenin was standing in a flea market for a while with a price tag of $150000, but now it's on sale for $250000 (inflation, you know).

I hope one of these New Russians will be able to shell out a quarter of a mil and bring the Grandpa Lenin back to Russia, where he belongs. Lenin succeeded in building a society of equally poor people, and it should be pretty upsetting for him standing in a city of unequally rich American people.

If 90 years ago one of the poor people returning from Moscow would say "I saw Lenin" , he'd be the only game in town. People would invite and treat him just to listen to some stories about this idol. So here I am, back from Seattle proudly announcing: "I SAW LENIN!"

And Pedro said, “Move over, Ravi!”

What first comes to mind if someone mentions Mexico? Beautiful beaches in Cancun with loaded Margaritas and hardworking people working in the USA in construction, landscaping and other non-attractive jobs. What comes to mind when someone mentions Brazil? Football, Copacabana, and that *garota de Ipanema*.

I was pleasantly surprised when a friend of mine working for a Fortune 100 firm told me that they used to outsource software development to India, but now they switched to Mexican programmers. Why? Cheaper. Way to go, Mexico! Good for you!

Brazil is yet another emerging supplier of programmers.

But is it right to work with a country just because it offers the cheapest prices? I do not think so. The software development outsourcing may be efficient only if you cherry pick the best of the best as opposed to hiring teams for cheap. Work with individuals, not with teams!

Let's apply the 80/20 rule one more time for an offshore team of five that charges $100 a day for a programmer. The daily offshore payroll is $500.

Such a team needs a local full-time manager here in the US, which earns, say $600 a day. We've got $1100 a day just for the payroll. Four out of five offshore programmers will definitely under-perform to say the least.

The better model is to hire one excellent offshore programmer for $200 a day, which will need two hours a day of the local manager's time (another $200). In this case the daily payroll is $400, and the chances of delivering your project on time will dramatically increase.

This formula won't work if your IT shop is run by a mediocre manager who does not ant to innovate and goes by the flow of the offshore partner given from above.

Visiting an offshore training camp for programmers

I wrote this while sitting in a training camp called Genghis Khan, a school for the rookie offshore programmers. The camp is located in the remote mountains of northern Mongolia. Many wannabe offshore developers come here for training. Since I am a popular personality among the Mongol Java crowd, they let me sit in the class where Baichu, the guru of offshoring was delivering a presentation called “Dealing with Overseas Employers 101”. These are my notes from this class.

1. America is rich, we are poor. It’s not fair, they have to share.

2. In the beginning, their manager will try to scare you by promising that he’ll check up on the status of your assignments daily. Do not be afraid – a status report is just a formality, and they’ll take whatever you write.

3. One of your major problems will be "what to write in status reports". Never write “I could not do it ” there. Americans like positive statements. For example, let’s say you’ve got an assignment to create a reusable component that will identify the number of failed database requests. You do not even have a clue what are they asking for.
The first week you spend on Google in hopeless attempts to find such a component. The status report for the first week should read “Comparing

various approaches of creating reusable db-failures component to find the most efficient and effective way for its development". During the second week invent something similar. Hopefully, on the third week something more urgent will come up and you'll get another assignment.

4.   Be prepared to spend the first couple of weeks waiting for the logon id and password to your employer's network. After obtaining these credentials, you'll find out that you still don't have access to a dozen of the servers, which require Unix logon. Your remote manager will promise you to resolve it as soon as possible, but because of the service level agreements (a.k.a. SLA) with the Unix support team , you won't get access for another week or so. Typically, it'll take about a month just to get you connected.

5.   Never say "I do not know". Accept all assignments – one of two things will happen – either you'll figure out how to complete the assignment, or it'll get cancelled.

6.   In conversations with your overseas teammates, always require detailed written specifications for each small program modification. Ignore their statements "I'd fix it myself faster than writing detailed specs for you". They have no choice and must work with you to show that your team is useful.

7.   Use the time difference to your advantage. For example, if you want to send an email asking for some clarifications, do not send it in the morning, because you may get an immediate answer. Do it in the evening (your time zone), before leaving the office – you'll get the answer next-day.

8.   If you have a choice, avoid fixed price projects. Hourly-based pay will allow to put a couple of extra hours here and there, and having a couple of extra tugriks or rubles never hurts.

9.   Experienced offshore programmers - usually do not try to obtain US working visas to work onsite. They'd be working a lot harder in the foreign land. It is not worth the trip.

10.   Always be polite – it'll get you far. Insert "Excuse me", "Thank you", "Yes sir" in every other sentence. Always smile - even during phone conversation.

(Then he showed this popular movie about an offshore tech support: http://callcentermovie.com/ .)

11. Change your local employer every three months. You are gaining experience daily, and even if the new job offers just a one percent salary increase, go there. It's a golden IT offshoring era – use it while it lasts! Or as they say, it's time to make a quick buck!

For some time I was speechless after hearing all these advices. Baichu spent at least half an hour after the class answering specific questions from students. I also asked him, if he really believed in his teachings. He smiled at me and said, "Welcome to the real world, ma'man!"

Disclaimer. This did not actually happened in Mongolia, this was just a fruit of my sick imagination - it happen somewhere else.

Lack of management in outsourcing

Some time ago, Information Week Magazine published an article "Outsourcing to Win: 4 steps to An Effective Strategy" .
They write, "Campbell says most companies lack the necessary management structure to effectively manage a team of outsourcers. With businesses spreading work around to more and more external contractors, they need to elevate vendor management to the executive level and hire the equivalent of a chief sourcing officer, she says. "There are not a lot of individuals within a company that have the multidisciplinary skills required for that sort of role." This is probably the only useful advise that the article gives – the rest is the typical gibberish of standard phrases taken out of real world context.

Liz Campbell hit the nail on the head. In my opinion, the lack of management is one of the two MOST IMPORTANT things that make outsourcing fail (I'll mention the second one a bit later). Of course the roots of this lack of proper management issue lay in the fact that in many cases the project manager has no say in WHO are their outsourcing partners. They are given from above and have roots in golf courses and fancy restaurants.

OK, the project with outsourcing begins, the project managers quickly understand that the low Bangalore hourly rates is a joke, given the fact that because of the typical corporate infrastructure issues (here in the USA), an offshore team of five $20p/h spent two weeks doing nothing but exchanging emails trying to get access to the proper servers. Let's do the math: 5*$20*8*10 = $8000. This money just evaporated from the project budget during these two weeks of doing nothing.

Interestingly enough, the upper management will never know about it because…and here comes the second main reason of why outsourcing is much more costly – CONSPIRACY. Project managers will cover up such losses because they do not want to disappoint their managers – their own yearly reviews and bonuses depend on success of their projects, so they'll quietly absorb these losses by asking their local team members to work more and tie up loose ends.   The local team members have their own reviews too, so they'll work a bit harder as well.

The article suggests to get familiar with the contract laws and religious adherence to standards. This does not come from the real world.

The next advice of this article is to move to team-based outsourcing projects. I think it's dead wrong. The majority of the software developers do not produce much. My advice is to work with the individuals that your team leaders personally pick. Introduce the competition within the offshore firm. Provide PERSONAL incentives for individuals. Some of the offshore teams are sand castles. People there can easily quit their jobs if someone extends them an offer with 2% salary increase.

And if your outsourced project failed, stop blaming Indians – just look at the roots of this failure at home.

Me goes to America!

The world is not flat, the quality of life is different in each country. While some people are happy with what they have, and if the economic conditions in their home country turn gray, they complain but try to survive. Other people start looking for better opportunities for themselves and their families elsewhere. They can relocate within the same country going places with better job options, and some people are not afraid to leave their own countries in search of that promise land. One of the most desirable destinations for people of all around the world is the United States of America.

A line by American poet Emma Lazarus describing the Statue of Liberty, appears on a plaque at the base of the statue, which ends with the statue herself speaking:

Give me your tired, your poor,
Your huddled masses yearning to breathe free,
The wretched refuse of your teeming shore.
Send these, the homeless, tempest-tossed, to me:
I lift my lamp beside the golden door.

And here I am, knocking at this golden door.

What a country!

A Ukrainian-born U.S. comedian Yakov Smirnoff has a number of jokes that can fall under umbrella "What a country!" I'm not trying to be funny here, but has my own list of things that were unusual to me in the USA.

I have arrived in the USA in January of 1992 from a grim society, which is now called "the former Soviet Union". Sarafanov's family gave me a place to stay and the food for free, and I'll be always grateful for that.

There is a big difference between foreign programmers who live in the USA and those who don't. How do I know this? Because I've started my career in the USA as a foreign programmer myself! The next several pages are not exactly about programming, but rather about my observation of American lifestyle.

I'm a USA citizen for many years. I always wanted to live in this country and thought I knew a lot about it, but there were things that seemed unusual during my first years here. This is my list.

- "How are you?" is not a question and you do not have to answer it.

- You can spice up any word by breaking it in the middle and inserting the word f..king there. The classical example from Pretty Woman is Cindef..kingrella.

- They have kneeling buses that can lower the door – disabled passengers are not second class citizens here

- Having good looking teeth is very important

- People can have different opinions on the same subject and remain friends

- If you see audio speakers at WallMart for as low as $10 this does not mean that the speakers cost as much as $800 are good. There are also speakers for $8000 and for $80000. What a country!

- Computer programming is the best profession in terms of return on investment: six to twelve months of training can secure you job paying $50K+ a year

- If a patient is diagnosed with a terminal disease, he'll be the first one to know about it, not the relatives

- Divorce is extremely expensive for men

- America has way too many lawyers per person

- Having an American passport is not enough to prove your identity while applying for a renewal of a driver license - it's only four points, but you need six to get the license

- If you are driving a car and someone hits you from behind, it's a good thing

- Buy one get one free deals: If you buy what you don't want, we'll give you what you don't need for free

- If an interviewer asks you to evaluate your skills on the scale of 1 to 10, add two-three points to your honest assessment – the person who asks will subtract them thinking that you've overestimated your skills

- If someone greets you with "How are you", the only right answer is "Great"

- If someone greets you with "What's up?", the only two right answers are "Nothing" or "Not much".

- The word "friend" is not actually a friend. The "close friend" is a friend

- Men change shirts every day (it may not be a fresh shirt, but it must be a different one)

- You can't and shouldn't try to bribe the policeman that is writing you a ticket for traffic violation

- All toys are made in China

- Americans call soccer what the rest of the world calls football

- In America, an English speaking country, you have to press 1 on the phone if you want "to continue speaking in English"

- The American dream is not to buy a house, but to have money for the down payment for a house and then being able to make monthly payments

- In December, you can see a Christmas tree next to Hanukah candles in every store – business is business

- People are the same in every country, but Americans are lucky to be born in the society and smart enough to maintain the society that is very close to the needs of a regular Joe

- Henry Ford said,

  If you think you can do a thing or think you can't do a thing, you're right

  This is my favorite quote.

My H1B story

I arrived to the USA with good experience in programming. I knew C and C++! Pretty quickly I got an offer, but my prospective employer would hire me only if I'd have a work permit.

There are several visas that allow you to work in the USA. So-called H1B visa is by far the most popular among software developers "temporarily" coming to America, which is short in enterprise IT talent. Visit a corporate cafeteria in one of the largest American cities, and you'll get a feeling that you are in India, even you are of Indian descent.

If you'll take an IT group of ten developers in New York, San Francisco, Chicago and the like, it'll consist of six Indians, two Chinese, one Russian, and, possibly, one person who's first language is English. You may accuse me of profiling, but this book is called "…without BS", remember? Welcome to the real world! The situation in the mid West is different, but the majority of IT shops are not located there anyway.

At least a half of these people are either still working under H1B visa, or have already received their permanent residency in the USA. The government tries to deal with the shortage of people in different professions, and H1B visa holders can work for six years unless they convert this visa into a permanent residency (a green card) or become US citizens in some other way.

After spending about three months in the USA I passed a technical job interview at a New Jersey company called InterAccess. They liked me and said that now I had to get my visa. I applied with a lawyer and after about three months of living with my fingers crossed, I got it. This is a long process: the prospective employer had to place an advertisement in the local newspapers that they were looking for people with my skills and that they should somehow prove that I was one of the best candidates for this job.

There is something called "prevailing wage", which means that that employer is supposed to pay me the salary with what is similar to what they pay other people with my qualifications. I do not know what my lawyer did, but if you'll watch this video http://www.youtube.com/watch?v=TCbFEgFajGU it may give you an idea how this process works internally. It's not kosher, as most of what lawyers do, but their main goal is to get the job done for their clients.

I got my visa, and my first salary 1992 was $33,000 a year, which back then was a huge amount of money for me and my family who just arrived. I didn't have any money at all – some cash from delivering pizza. In fact, when my wife and son arrived, I had seven thousand dollars in debt, which I borrowed from friends and relatives. My wife got an H4 visa, which did not allow her to be employed. She started cleaning houses of rich people in Staten Island, NY. During the first eighteen months of my day programming job, I was also delivering pizza on Friday nights and Saturdays.

My long term goal was to became a citizen of this great country, and the first step toward this goal was starting a "green card process" that should have been done by my employer. My first employer was a small company and they did not have much experience with green cards, so I found a large consulting firm called Trecom Business Systems that transferred my H1B (this visa allows you to work only for a specific employer). This firm paid me a salary of $45K a year, which was meant a major improvement in my life style. This is the last time I quote my income. Why? I'll explain this a bit later in the section "What's your salary".

After six months my new employer applied for the green card and this process took two and a half years. All of these years I have been working hard for this company. I knew that people with my skills were getting better pay, but this didn't bother me at all, because I felt that this was a fair deal. I wanted a green card sponsored by my employer, and they wanted my skills for a bargain price. This was a fair game.

In 1996 I became a permanent resident of the USA. My dream came true! As soon as I got my green card I created a one-man corporation and started to work as an independent consultant. Sorry Trecom, but we've been helping each other for three years, and it's time to move on.

Are H1B workers abused?

Most of the H1B visa holders are not the people who occupy the leading positions at Google or Microsoft. They usually work for consulting companies, which send them to their clients to work on various projects.

Often (but not always) these consulting firms pay H1B workers a smaller salary. Today's prevailing wage for the H1P Visa candidates is probably about $55,000 a year, which is close to the lower side of the programmer's wage range.

If an employer gets a good guy for $55K a year and charges the client $90 per hour, it does not take a rocket scientist to figure out that there is a markup there. Sure, consulting companies pocket a big chunk of cash on H1B Visa developers, but as I said above, it's a fair game. I've been there, but I did not feel abused.

Now let me put on a different hat – these days I often hire people. I try to get the best possible person, with the best skills for the amount of money that make sense to our firm. But I also try to find them interesting work. In the beginning, these people don't deliver. In the beginning I may need to teach this person a technology that we are interested in at the moment, and I want to keep him on board longer to return our investments in this new hire.

Will this guy stay working for our company? If he is an H1B Visa holder, the chances are a lot higher that this guy will stay working for me for several years then if I'd hire a green card holder or a US Citizen. You maky not like what I write, but this is a decent deal for both parties for newcomers who want to stay in the country and for employers who want to get the job done and make a profit.

There is a need in H1B software developers, and it'll just grow. There is a programmer's guild (http://www.programmersguild.org/) that calls for limiting the amount of these visas given to foreigners. They publish videos and horror stories showing how rotten the system is and the "unfair" things that lawyers do.. There were different attempts by the government to compensate Americans for every foreign worker that they brought into the country. The government forces every employer who is inviting an H1B worker to pay a certain amount of money (I believe it's above $5K) to the government for retraining American citizens. I don't think that any of these measures will change the overall picture as long as the company that is hiring an H1B workers makes a profit. These hefty fees will not help Americans in retraining unless they really want to get retrained.

No, foreign programmers are not being abused by capitalist sharks. If you are an H1B software developer, do not behave as girls who were brought into the USA with a promise of dancing on Broadway, but now dance on the stages of Go-Go bars. With H1B workers, each party knew the rules of this engagement upfront and is moving toward achieving their goals.

Have I taken your job?

The green card allowed me to work for any employer, and I was on the market as soon as I got it. The IT job market is very competitive. If you are good at what you are doing, you'll find a better deal. If your skills are a little rusty - sit down and shut up.

Once in a while I run into an article citing people who had lost their jobs and could not find the new ones. These people were born in America and accumulated lots of year of work experience. One of such articles published a story of a guy who was working in IT for thirty five years, and after sending about a hundred résumés couldn't find a job.

I feel sorry for him, but there are other reasons why such people can't find jobs. If you spend thirty five years in the industry, you are over fifty. Of course the age factor plays a role in the search for employment but in the United States enterprise IT industry this is not a showstopper as long as you perform.

I have seen job ads in Russian media, which explicitly state that you have to be under thirty five years of age, which is industrial-strength stupid. As if before thirty five your brain works fine and if you are any older your brain is fried and does not perform. This is absolutely wrong, but is none of my concern since I don't live in that country and am not planning to.

In the USA, if you are good software developer your age doesn't matter.

The articles with complains don't usually mention that people who lost their jobs do not want to move from their home towns because *they have roots*.

What a BS! If you can't provide food for your family, forget about roots, pick up your butt and relocate to the area which has a demand for people like you in two weeks.

Are you moving where the job exists or you'd rather find several "valid" reasons why you can't go there? If this is the case, do not complain that I took your job. A Free and open market is one of the main values of the USA. Let's compete.

There are lots of people who are working under H1B and are ready to rock and roll! They are ready to start working on any project literally tomorrow at any point on Earth. Many software developers from India live in so called *guest houses*. Their consulting companies are renting an apartment and put there two or three people in a room. They also share a living room, kitchen and bathrooms and are OK with it. They buy inexpensive cars and car pool to work. Technically they can go anywhere they want; they are very mobile.

Don't like these intruders? Then build a fence around the country and do not let anyone in, but be prepared to take all these low-paid jobs that programmers from guest houses do. And stop calling yourself a free nation with an open market economy.

These articles don't usually reveal that people often just stay in large firms for decades assuming that nothing would ever happen to their employment regardless of competitiveness of their skills. I remember working on a project for a large telecommunications firm. One of their aging programmers was literally doing nothing at work. He had a strange screen saver: large 3-D digits would rotate on the screen showing a different number each day: 254…253…252…

When I asked about the meaning of these numbers, he readily explained that this is how many days left to his retirement, and after working in the same place for more than thirty years, it would be more expensive for the firm to fire him with a large package for doing nothing, than turning a blind eye to his lack of productivity..

I work as a consultant for various corporate clients. Recently, I had to re-engineer an old mainframe-based system to a more modern service-oriented one. I spoke with software engineers and managers and one of my questions was if they were willing to get retrained and learn new technologies. People were not too keen on doing this. One of them gave me the following answer, ″For this new project we will hire offshore consultants and we'll manage them".

How do you like this attitude? I am happy with what I have, I have this cozy nice little place at work, leave me alone and don't tell me to get re-trained. I am fifty years old for crying out loud! I want to manage offshore developers, and I want it now!

They do not realize that the situation may change rather quickly, and these people from overseas may take their jobs and kick them out of their comfy cubicles and offices. If you do not want this to happen, compete with them. You spent fifteen years in this company? Great, this means that you have excellent knowledge of the business, and this is something that offshore guys don't have – this is your edge. Keep your technical skills up to date and no one will beat you! Before blaming all these guys from overseas, ask yourself, "What did I do to secure my job?"

America is a great country, and who made it great? Immigrants. Even if you were born in this country, your parents were immigrants coming for a better life. By the way, stop complaining about these illegal. Your ancestors came here legally a hundred years ago... just because America was accepting everyone legally back then. It's neither yours nor their achievements. Mexicans who are coming to America illegally do so to help their families to survive.

The first generation of immigrants in the US were fighting for survival, they were super achievers. Yes, I'm a first generation immigrant. I'll stay focused and will work harder, because I started in this country from scratch being an adult and need to catch up.

I do not know who was the first to make this statement, but it's correct. Enterprise software developers create applications for business users "to make their lives easier". What a cliché! First, someone decides that there is a need to make the users' lives easier and finds the money to fund this exiting process. Then, the users need to explain software developers what exactly makes their life difficult. This explanation should be done in a form of functional specification a.k.a. functional spec.

This is when the process of pulling teeth begins. The problem is that often business people can't formalize the existing process so we, developers can show them the way to the bright future. The other problem is (especially with legacy applications) that sometimes it's not easy to find a person who knows the entire workflow of the application. This person is either quit, retired or died a couple of years ago. The third problem is that business users are busy or want us think they are. We can't bother them too much because it distracts them from fulfilling their duties.

If you are a software architect or a team lead you start with asking questions so you can understand what the users think they want from this stupid computer. Then you might even suggest your users conveniences they did not even think of due to lack of understanding of capabilities of these smart computers.

That's why having a good interviewer with deep understanding of technology is one of the main ingredients of the success of the project. When the project finally goes live (I've yet to see one that goes live on time) , this application is not exactly what the users initially wanted as per that short functional spec. "Oh, by the way, can we also squeeze in this little feature?"

How to select a software vendor for your next project

If you are a development manager of an enterprise IT team, once in a while you'll have to deal with IT vendors – consulting companies that you hire for a specified period of time just to help with a particular project. Our company, Farata Systems is such a vendor company. We consult on various projects for large and small enterprises. At this point you may think that I'll start bragging about how great we are. On the contrary, I'll tell you about our failures and the lesson learned.

During the last year our company has lost bids on a couple of consulting projects, and we've noticed a pattern there. Here's one of these cases.

A large company approached us asking to bid on a large project. We knew how to do the job, we've estimated the time and resources and came out with the numbers: $200K over six months. In about two weeks we were notified that they've decided to go with a different vendor that offered them to do this job for $60K. We shrugged and moved on with our business.

Six month later, we've got a call from them asking if we could help with that project. By that time they paid already $300K to the vendor-winner, and the project was not finished yet.

Since we've seen similar scenarios in the past, it seems like a sales pattern – some vendors are giving unrealistically low estimates just to get the foot in the door. Then, little by little, they present valid reasons that require additional budget. The client is on the hook because someone's career is at stake, and they have no choice but sign the next invoice and keep dragging the project until it either produce at least something that works or comes to a full stop.

A little lie during the job estimates seems to be a trick of the trade of many salesmen across the industry.

When an enterprise IT department gets the budget for a new software development project, they ask several vendors to bid on the project. Some of them will come up with beautiful PowerPoint presentations on the current state assessment of your system, the future state assessment, and a roadmap to this bright future, which can be fairly technical.

More agile vendors hate this BS paperwork - they present you a two-page write-up containing a technical solution addressing your functional specification. The third type of vendors is prevalent, and they will present a decent slide deck and some technical meat.

Then the cost comes into picture. Vendors’ marketing people will estimate the cost of the resources. If you are a newcomer in the enterprise IT, you might not know that “resources” are actually people. Salesmen do not call software developers people, they call them resources. In one of the corporate meetings, I've heard an account manger saying, "A father of one of my resources died so this resource will not be available for a week". Could it get any worse? Actually it could, for example, "An ancestor of one of my resources died so temporarily it won’t perform its functions".

If a cost estimate of one vendor is substantially lower than others, they are either not telling you the whole truth or will be using dirt cheap resources.

So how you, the development manager can pick the right vendor that will deliver the project in time (or close to the deadline)? Here’s a solution: give each vendor two weeks and ask them to come back with a working prototype of the system to be developed. Important: you have to pay for this two-week job to each vendor.

I want to make it crystal clear – they should come back to you not with a nice-looking diagram of the system, not with the wire frame created with some prototyping tool, but with a working application that is built using the software approved for the project. Of course, this application won’t be fully functional, it’ll run locally on a laptop with lots of dummy pieces of code, but IT HAS TO WORK.

Say, you have five vendors bidding on your project. The odds are that after such a two-week offer at least two of them will quietly withdraw their proposals. The remaining vendors will present their working prototypes on time and, all of a sudden the job of picking the right vendor becomes easy.

Yes, it’s cost you a little bit of upfront money, but it was the money well spent.

This was our lesson learned, and now we often offer our perspective client to try us out on this mini two-week project. It’s fair to our clients, and it’s fair to us.

What’s your salary?

In the USA, your salary is the most confidential information. Should you even answer the question about your salary? Never. People who are entitled to know this number already know. Only your boss, HR and, sometimes your spouse know this magic number. There’s an unwritten law: never ask your colleagues or even friends how much they make. There is another law: do not brag about your salary trying to impress people. There are a several reasons why this is never a good thing to do. Let’s consider some use cases.

Use Case 1. Joe and Larry are colleagues working for the same company, they have great relations and perform similar duties. After having a couple of beers at the corporate party, Larry says, “Joe, you did a really good job at this tough project with unrealistic deadlines. I hope they’ve compensated you for all these long hours.” Flattered Joe says, “Yes, I got an extra $10K as a bonus”.

Larry response, “Wow, this is really cool”…and gets upset and pissed off. Two months ago he’d been working really hard on a similar project, but did not get anything other than “Thank you”. Larry thinks that it’s unfair. The climate in their team slowly gets worse and Joe and Larry do not go for a beer any longer.

Use case 2. Many years ago Sandy and Max have graduated from the same college. Now they work for different firms but their families have friendly relations and often spend time together. During one of such gatherings, Max asks, "Sandy, we are both programmers with similar skills. I'm thinking of making a move with my career. What's the realistic salary I should ask for?"

And Max makes a big mistake by saying, "Well, you know my skills, and I'm making $80K working on a time tracking application for our Human Resources department." Sandy spent years doing his Ph.D. research and works now on complex algorithms in a lab funded by the government. He makes $60K a year, which is good enough to make ends meet.

After the party, Sandy's wife Mary found out about Max's eighty grand a year. From that day she could not sleep at night. How come? Max has hardly graduated, while my Sandy was always a genius…Sandy and Mary got divorced in a year.

Use case 3. A firm puts an ad with a job description that fits your profile advertising a salary range $70-$80K a year. Your current salary is $68K, you feel that you are underpaid (even though there is no such a thing as "underpaid" people) and decided to apply for a job. You meet with an HR person of this company, and they ask about your current salary. You should answer: "I'm looking for a more challenging job, and the position you're trying to fill is exactly what I want, I'm qualified for this job, and will be happy to work for you for $80K a year".

Do not reveal your current salary. If you need to fill out a job application form, leave this field blank. Most likely you'll be able get away with this. If not, oh well, look somewhere else. Most likely they won't pay you eighty after figuring out that you're making sixty eight. This employer knows the qualifications required for this job, they are well aware of prevailing salaries paid for similar jobs by other firms, so their only concern should be if you can do this job and not how much money you made before.

Use case 4. You are looking for a job, and your recruiter sends you to a client company for a job interview. He knows your salary requirements. The potential employer asks you about your salary… Do whatever it takes but don't not answer this question. I do not know who said this first, but it's a golden rule of salary negotiations:

The first person to mention the number loses

Remember, salary is just a part of the compensation package. Here's your mantra: "I'm open for any fair offer. If you decide to hire me, I'll gladly consider the entire compensation package rather than arguing about the base salary". Usually, your agent will be able to negotiate a better deal for you. Let him do his job based on his good relations with this client.

Use case 5. You are a successful software developer, your salary grows faster than average in the nation. You've managed to increase your salary from $60K to $120K within the last five years. This is quite an achievement. You feel important and want to teach other people how to manage their careers. Your body was craving for a couple of Wows, and you've shared you success story with your neighbor. He said, "Wow"…smiling to himself as he was consistently making $150K during the same period of time. Luckily, your neighbor would never reveal this number. Your self esteem is saved.

Use case 6. Alex and Lisa were going out for two years, and finally Alex decided to propose. By some idiotic rules young men often purchase expensive diamond rings to impress their wife-to-be and her friends. Alex has purchased a $15K diamond ring, and proposed to her. She was so happy and said, ”Yes”. In a couple of months, she asked Alex about his salary. She caught him off guard and he said,”$60K a year”. Lisa was a student majoring in accounting. She quickly put two and two together and decided that she couldn’t live with a man who can spend $15K on a ring having such a modest salary. I do not remember if she has returned the ring though.

Use case 7. Marsha is leading a small team of software developers. Marsha is in her 40s, and she's been with the company for 10 years. Ashish, 27 is one of these sharp god-like gurus who programmed 80% of the project. When Ashish found out that Marsha's salary is $90K and his is only$60K, he decided that it's not fair and immediately started looking for a new job. He did not know that only because of Marsha's connections within the firm was your group given this project in the first place, and because of her Ashish always had the green light when the business analysts input was required. Yes, Marsha was not as good of a coder as Ashish. So? Often team members tend to overestimate their contribution to the success of the project. And you as a project manager better make sure that they do not start comparing their compensations.

Use case 8. Somehow Gordon figured out that Frank is making more money than him. During the annual review, when he found out that his salary will increase only by 4%, he said to his manager, "How come, I am not worse than Frank but he makes more than me".

Yakov, the manager, said that he'd take a look at this situation. In about a month, he invited Gordon in the office and said, "Our department is not doing that great, we've had a series of budget cuts, and even though you are a great asset to our group, I'm going to have to let you go. Sorry, man!" Of course, this was not the real reason: Yakov did not need this bad egg on his team.

Never use someone's salary in negotiations of your raise

You think you are underpaid? Update your resume and hit the job market. Can't do better than now? Sit down and shut up. It's a bit rude, but welcome to the real world.

Americans neither ask nor answer questions about salary. But if you run into a newcomer who may ask you directly
“What’s your salary?”
Just say, ”Even my wife does not know this number”, and give them an American smile.

Underpaid? Quit!

Is there such a thing as underpaid people? The short answer is, ”No”. Each person is getting paid what s/he deserves at this moment.

You may think that they should pay more for your hard work. I have bad news for you: you are always getting the right compensation.

You feel that you worth more? Hit the road, Jack and see if there is anyone else who also thinks so. Try to find greener pastures.

Let’s say your current annual salary is $70,000. You just got an annual review and a raise to $74,000. An extra four grand of your gross income translates into twenty seven hundred a year of after tax money $225 a month.

If you are not too happy with this kind of raise, try one of the following two strategies.

Quitting strategy #1. Do not argue with your boss. Just hit the job market. There are two possible outcomes. Outcome number one: you’ll find out that even $74K is way too much for your skills today, and the best offer you can get is $69K. But the negative result is also a useful result. Keep your current job, improve your technical skills to match the market demand. Just sit tight, study to get your technical skills current and six months later try to hit the market again to see if you can get a better offer.

Outcome number two, you’ve got an job offer for $80K. Accept and quit your current job.

Quitting strategy #2. If you believe that you can make more, explain to your boss why you think so. Give her a chance to re-evaluate your compensation. Wait for a couple of months, and if nothing happens, use the quitting strategy #1. If someone makes you a better offer, quit without thinking twice.

The chances are that when you give your two week notice (the industry standard), your current employer will offer you to match or even beat your new deal. And remember the golden rule: don’t accept a counter offer.

Can't find a better paid job in your town? Move to a different one. Can't or do not want to? Then do not complain - there is no such thing as underpaid people.

Overpaid? Hardly

The Information Week magazine has published an article “Down to Business: Is Exec Pay Excessive?” The author answers, "Hardly". I agree with him because I believe in the open market.

If you are wondering why a policeman earns $50K annual salary while risking his life, the author explains,"...it just means that employers can find more of them at the pay they now earn". Do you think it's fair that a 20 year old basketball player whose vocabulary is "Yo, man" and 500 more words earns ten times more that the President of the USA? I think it's fair because thousands of people are willing to pay big bucks for a game ticket when this Yo-Man is playing. He earns his salary. Big time! Do you think that his agent is overpaid? Hardly. Without his agent this superstar would still be playing in his neighborhood playground on weekends.

Or take the tennis champion Maria Sharapova. Does she earn her pay check? Absolutely! She's pretty and technically she does not even need to play tennis. Maria can just wear these sexy Nike outfits, walk around the tennis court and scream periodically. Trust me, men will keep purchasing tickets.

Do you think that your boss earns too much? I do not think so. Do you think that without you the project would fail? Do not think so. Why your boss is getting paid a lot more without knowing Java or C++ like yourself? You boss has much more responsibility than you do. Your boss is responsible for all the mistakes you make. Your boss wears a Blackberry, and only stupid people can think that this is a privilege. Your boss is on a leash 24x7. Ask yourself, do you think it's going to be better if you get a new boss? Still do not like your boss? Quit, and see if the new one is better.

Do you think that they pay you too much? Hardly. You worth every penny you make today! If someone will pay you more tomorrow, you’ll be worth every penny too. But that is tomorrow…

Poor advice to laid-off people

I like reading blogs. People have opinions and I have opinions. Some of the blogs may change my opinion or two. But this time I ran into a blog that just gives poor (IMO or IMHO) advice on things that a person has to do on the first day after being laid off. At this point, kindly switch to Jason Kester's blog and read it. Done? My turn.

Being laid off is one of the most serious cataclysms in anyone's life. Stronger that this could be moving to a new house (been there) and American divorce (no practical experience here, just the horror stories from people who went through it).

If any person will tell you "I was laid off, but received good severance package. So I do not really care", s/he lies to you. S/he's very upset. Everyone should be upset receiving a sudden punch in the face.

Jason suggests to book a flight and see the world because "you will never have a better chance to see the world than right now". I love travel, but I prefer to do so when it's convenient for me, and not because someone kicked me out of the house as an Indian cow that stopped producing milk. It's kind of weird analogy, but I just came back from India , and I went there not because of being laid off.

Then, Jason makes another cute statement, "You have a pile of savings and a severance package". My sincere congratulations to the author, but I can tell you a little secret, most people (in the USA) neither have a pile not a package. The only thing they have is a burning painful thought, "How long can I last till they kick me out of the house". Because these days American dream has converged from having a house to making money for a down payment to a house.

One more gem, "You've got 6 months to a year before your skills start getting rusty". If they are not rusty, why not getting another job, and see the world when it's going to be your choice, not their?

Moving along…Jason recommends to fly to Thailand, and this is not a bad idea, if you go there when the sun is bright and the grass is green, which is not the case on the first day after the layoff. Contradicting to himself (remember the pile of cash?), the author states that Europe is crazy expensive, and you may even consider going to South Africa, where you get a room for $0.75. After years and years of travels, I've learned one thing: hotels cost pretty much the same everywhere in the world if you'll compare apples to apples. Go online, take any hotel chain you like: Hilton, Sheraton, Four Seasons and pick any country in the world. You may be surprised to learn that the prices are the same.

Of course, college kids happily travel in Europe staying in Paris hostels for $10 a bed. But there is a big difference between the state of mind of happy campers whose main goal is to quickly get drunk and get laid (without off) with a different person each day, and a laid off guy/gal with a little pile in their hands.

Then Jason starts bragging, "Did I mention that I take about 9 months vacation a year". Congratulation again, but the rest of the world are just not like that. Poor thing, he "start missing work after about six month away". I feel for him. Staying in $.75 rooms for six months in a row must be worse than being an inmate in one of the American prisons. Jason reveals that he spends $1000 a month staying away from developed countries. Thirty five bucks won't get you far, but you may try one of the Frommers guides like Scandinavia on 40 Dollars a Day (get it for a penny at Amazon) . Oops, it's been published in 1987…Sorry. With today's weak dollar, even Rachel Ray stopped telling her stories about $40-a-day travel and switched to a safer 30-min meals show.

To keep your skills up to date, you should take your laptop with you. I wonder if these $.75 rooms offer safe deposit boxes that can accommodate your laptop while you are out enjoying $3 lunch?

I can keep commenting, but the bottom line, that blog gives a BS advice. Hopefully this book will help you to manage your career and be ready for layoffs, which suck. Really.

Is life in startups any different than in corporations?

Vacations are meant for reading. This time I've picked an e-book "Eric Sink on the Business of Software". This section is not a review of this good book, but rather my own thoughts and comments inspired by reading what Eric thinks about running a small software development company.

Below are some *quotes from Eric's book* with my comments

- "*I like the smell of a freshly killed bug."* Very well said. I'd take it one step further and submit to Wikipedia the following definition of a geek: "Software geeks are people with a smell disorder. Most of all they love the smell of a freshly killed bug".

- "*Good communication is not 50% listening and 50% talking, It's more like 80% listening and 20% talking*". It's very true. Can you listen to someone other than your boss or wife without interrupting for more than three minutes? If yes, you have good communication skills.

- "*Your ideas are worthless*". Exactly! This is long and winding road between your great idea and a PRODUCTION QUALITY software product.
- "*The purpose of 1.0 is to help pay for the development of 2.0*". This means that you should not try to put too many features into the first release of your product – get it out the door and start making some money.

Eric's message about not using your house as collateral for your business is not strong enough. My message is this: "Do not even think about it!"

One day you came home telling your wife, husband, or a domestic partner that you've got an idea about developing a software product that will change your life and make you rich. But to implement this idea, you need some cash – let's take a second mortgage to get this cash from our house.

The only proper reaction from your wife, husband or a domestic partner is showing you a middle finger.

No, there can't be any special circumstances that would justify a chance to lose your house, if you made a mistake. If this idea is that great, why don't you try to convince anyone other than your wife, husband or a domestic partner to shell out some cash for implementing it? It's not easy? I know. But maybe this means that your idea is not that great?

Fine, if you still believe that this software product is your future, go ahead – start spending extra 5-6 hours daily (after your day paying job) developing your product, or get a second job if you need to hire a software developer.

I had and still have lots of ideas, but over the years I came up with the following cold-shower technique. Let's say, you have an idea of THE software product. Assume that you've already created it. Fast forward the time machine and visualize the day when you've completed development of your one-of-a-kind-state-of-the-art software program. Congratulations, but what's next?

What are you going to do with it?
Can you sell it to at least one person? Price does not matter.
Can you make anyone pay even $10 for your creation?
How are you planning to advertise it?
Do you have even a slightest idea about marketing of your product?
Do you have the budget for marketing? Your Web page explaining the revolutionary effect of using your newborn baby will go unnoticed unless you will be constantly promoting it.

Eric's book is about creating a small Independent Software Vendor (ISV) company that creates profitable software. He mentions that there are companies that do both – develop software and offer consulting services, and this is how his company has started.

I'm also partner in a company that does exactly this – we develop software components and offer consulting and training services as well. This business model allows us not to carry all our eggs in the same basket. This model has the following advantages over a product-only ISV:

1. Having two sources of income (consulting and product sales) is better than one. This is a no-brainer.

2. Both of these business activities complement each other dearly:

a) the money earned by consulting or training gigs can be used for the development of a software product. Such internal investment is a lot more attractive than asking for venture capital (VC) elsewhere.

b) it's a lot easier to get consulting gigs if you are also an ISV. The fact that you can develop advanced software adds a lot of credibility to your consulting services. This really helps us to stand out from other consulting body shops that bid on the same project.

3. If our software product sales won't be as good as predicted, we won't need to turn off the light in our business. Armed with the knowledge gained during product development our technical skills (so needed in the consulting business) are always up to date, and it's pretty easy to demonstrate.

Startups may have different problems than larger corporations. Employees in startups often work harder and have to wear multiple hats during a day, their compensation is lower and the dreams of capitalizing on a product they develop comes true for only a tiny number of ISVs. But hey, you never know…Maybe the startup you work for will become the next Youtube or Facebook.

A small ISV just can't afford hiring the wrong people (both employees and contractors). Wrong people are the ones that are either not technical enough or do not have proper communication skills. For example, we've had a technically sound worker who did not like answering emails. After several attempts to get the status of his assignment, we had to fire the guy. A larger firm would never do something like this, but a small ISV just can't afford to have such a person. Yes, it's a pity that we've invested some time into this worker, but we can't make success of our company dependent on the mood of one programmer (even if he's has good technical skills).

We had to let go of another person after about two months of employment – he was not good enough technically. Hiring this guy was a mistake in the first place, we've lost several thousands of dollars on him, but we must cut our losses quickly, learn from our mistakes and move on. Large corporations have lots of dead wood, which burns large chunks of their profits. Small ISVs should not tolerate this.

Eric does a good job explaining the difference between programmers and software developers, and he makes a very important statement, "a small ISV should not have any programmers". This statement might sound strange, but it's not if you realize that programmers are the people who just write code and do nothing else. Not to be confused with people called software developers, which do many other things like talking to users, making decision, perform testing and ARE CREATIVE.

Our small ISV company has people working in different countries, which makes having programmers (not developers) even a bigger no-no. Unless you can afford writing crystal clear specs for programmers, which small ISV can't, having programmers is expensive even if you pay them relatively modest salaries. They may not understand (and care) why they were asked to write this piece of code.

The time difference adds to the problem: you give them an assignment TODAY, and TOMORROW they respond that they did not understand it. It's a bummer. You've lost a day, which may affect your deadlines, and the salary paid to this person for yesterday's work was paid for nothing.

My short vacation was over, and I've completed half of Eric's book. I like the book, and may finish it one day. But I will follow his own advice – do not read one book on the business of software – read ten.

Why people work overtime

It's 7PM, and you can hear from your cube that people are still typing. Why they do not go home? Are they forced to stay late? Are they getting paid for these long hours? Why?

First, let's briefly touch upon the small startup culture – these people have some "idea", and who knows may be their little company will become the next Youtube or Facebook. These shops are either self-funded or operate under the scrutiny of VC. So when you hear that a company XYZ received $10M in venture capital funding, this does not mean that anyone in this company became richer.

This means that they earned the right to continue working on this idea/product envisioned by the founders of the company. People in startups work long hours, wear multiple hats, and hope that N number of options they have worth nothing today will turn into a fortune some day. Welcome to California! The startup spirit lives there. People think not just salaries, they think options (shares). If I'll just stay for two years with this company, I might exercise some portion of my options. This is the Silicon Valley's way of getting good software developers working long hours for sub-standard wages. Let's leave this group of people alone and wish them good luck. I understand their motives.

The East Coast is different. Unless you are in Boston, you don't know much about startups and options, and just work for one of the larger enterprises. We've already covered two ways of earning income: working as an employee or a contractor .

Contactors work for money. Period. If someone tells you "I prefer working as a contractor because it gives me more freedom", this is BS unless there are some special circumstances (i.e. you've enrolled in a PhD program or can work only a small number of hours per week).

Since contractors work for money, they like working overtime and they are getting paid for these hours. Some employers try to save a little bit and insert a clause in the contractor agreement stating that a contractor works a professional day at so-and-so hourly rate. Some smarty pants from HR came with this idea called "professional day", which means that the regular 8-hour day may be occasionally stretched to ten hours . If you've signed the contract that pays $80 p/h and on Wednesday need to work for 11 hours, you are getting 8x$80 plus another $80 for the eleventh hour, which is considered overtime.

Contractors have very simple and healthy relations with their clients. You need me for six months? No problem - $80p/h. You need me for two weeks? No problem. $100p/h. Why is it getting a lot more expensive? Because employers must pay for the convenience of having a skilled worker just when they need him on a very short notice and for a short assignment. You click on the button, and Joe is here. Seasoned employers understand that Joe-the-contractor will have some non-billable time after these two weeks and this higher rate should make up for the lost earnings. Besides, hiring a full time employee is like getting married, the wedding is expensive (you pay the recruiting agency hefty finder's fees) and divorce is getting even more expensive. Let's leave this group alone. I understand their motives.

Moving to the most complex case – full time employees working overtime. Let's single out the managers – these people are there to make careers, and they have to work long hours. These endless meetings steal their time and they have to stay late to get anything accomplished. Their higher-level incompetent managers give them unrealistic deadlines, and, if they are incompetent themselves, they just pass the pressure to their software developers, business analysts and testers, which start staying late once in a while. This is fine as long as it's happening once in a while. But all of a sudden, you found yourself working 10-12 hours every day without getting paid even for one extra hour.

Why people do this? I see several reasons:

1. Your technical skills are not up to date and you are afraid of losing this job, especially if you have no discipline in spending. Why did you purchase that car that you could not afford? Did you really need that apartment in Miami with zero down?

2. You are promised B-O-N-U-S at the end of the year, and if you won't be nice, you r B-O-N-U-S will become even smaller. October and November are the most important months for making bonus decisions, so you better behave. When you get this bonus, do a simple exercise: Add your salary and bonus and divide it by actual number of hours you've spent in your cubicle. You may be surprised by the results!

3. You are a workaholic and just like to work.

4. You have family issues and would rather stay at work than go home.

5. Your company pays for your college, and you have to show your appreciation by working overtime.

6. You are a smart kid, and working overtime gives you a chance to better learn the business you are in and improve your technical skills. You are planning to move.

7. Your technical skills are very poor, and staying overtime is the only chance to get even simple assignments done.

8. You or someone in your family have a disease and you need to have good medical insurance and changing jobs is not an option.

9. You need to have daily meetings with your offshore team in India, and because of the time difference you have to start working at five in the morning. Leaving from work earlier than your boss is not an option, so you routinely put extra hours.

The bottom line: be good at what you are doing, and then you won't need to say "Yes, Sir" every time when your manager decides that you are a second-class citizen and your main goal is to support his/her career promotion.

If you are good at what you are doing, you are allowed to say "No" and still keep the job. Pretty simple recipe, isn't it? You'd better put these extra hours taking some extra classes in your local college or self-study. You do not need to change your profession.

Do the same things as other people do, just do them a little better.

SOA, RIA and the Human factor

This section of the book is a little bit more technical then the others, but it delivers a message about important motivations for implementing any new technology in the enterprise.

While making a presentation on Service Oriented Architecture (SOA) I've asked the audience the following question, "What do you think is the driving force for implementing any technology or architecture in a decent size Enterprise?"

The answers were typical – better code re-usability, accessibility… But I was looking for a different answer that has nothing to with technical merits of any technology. Based on what I see in the real world enterprises, the main reason of implementing SOA or any other IT initiative are career goal of individuals working in this organization.

People want to become more visible and move up the career ladder. Implementing SOA across organization can make enough noise to move them to the next level.

SOA Ground Up

The SOA in your firm can evolve from the ground up. For example, an ambitious architect attends conferences, goes through training, reads white papers, and now he truly believes that SOA is the right way to go. He's aggressive and influential and manages to convince the CTO or CIO to allocate funds for this.

If you are not an architect but just a software engineer, you may try to start convincing your application manager, but an average application manager does not want SOA. He's busy with business as usual: do not forget to call into the change management meeting, fix yesterday's production bug, deal with an offshore team, attend 5-6 meetings a day… An enterprise IT manager is a firefighter. Imagine a firefighter that is putting out a fire…Here you come in a clean white suit offering to sell some new bells and whistles for a fire truck…

The chances of a software engineer to start SOA movement are very slim, really.

SOA Top Down

Another reason for SOA is this: your CIO went to a conference and attended a presentation of an energetic speaker, who clearly explained the benefits of SOA over any existing architecture. This is the worst case. Subordinates will not resist - they also have career goals…

Your CIO will come up with an SOA project plan based on available funds and resources, which means that this project is doomed. Your CIO will never admit that it was a mistake, which makes things even worse.

All application groups will roll up their sleeves, will work hard and meet the deadlines. But let me remind you a quote from a must read book by Fred Brooks called "The Mythical Man-Month":

An omlette, promised in two minutes , may appear to be progressing nicely. But when it has not set for two minutes, the customer has two choices – wait for it or eat it raw. Software customers have had the same choices.
The cook has another choice; he can turn up the heat. The result is often an omlette nothing can save – burned in one part, raw in another.

SOA as a burner

The third reason for implementing Enterprise SOA is to burn some cash: we (IT) have $2M – let's service-enable all of our applications by the end of this year. Why $2M? It's elementary, Watson!

You need to burn $2M this year, otherwise you won't get funding next year.

Get some high-price software products. Who cares that no real feasibility study was made? Just go with a well known vendor and get expensive software/hardware - your developers will figure out how to plug in that Enterprise Service Bus into your SOA.

SOA Maturity

As a consultant on SOA projects, I had to interview technical managers to figure out if SOA is a realistic solution for their company, and how the CUSTOMERS of this particular silo application will benefit from implementing SOA in the firm.

All these managers are working hard to fulfill their current obligations, to make sure that their applications are up and running in production, that support calls are answered in a timely manner, the offshore team is delivering, and on and on and on. And here I come in a white suite asking all these smart questions:

"How do you run business today?"
"What parts of your application are candidates for being converted into services? No, SOA is not the same as Web Services".

And these polite but tired people are looking at me, listening to me, they've seen already other architects trying to change the way they do stuff in their organization. Several similar initiatives have failed before, and here we go again…

I can't forget one technical manager – this lady was having hard times even finding time for meeting with me. During the appointment she was polite, but looked tired and worn out. She's a professional and was trying to answer my questions anyway, but her phone rang off the hook, he right hand was moving the mouse over the MS Outlook screen. "I'm so sorry, I need to take this call…" And I'm thinking to myself, "What am I doing there? Does she really need SOA?"

Is your organization ready for SOA?
Does your company have the right skills, infrastructure, SLA, SDLC in place?
What methodology are you planning to use to identify services?
Do you have lots of third-party shrink-wrapped applications?
You know how to wrap up your CICS application into a Web Service, but are your mainframe developers willing to get re-trained?
What about the governance?

Some architects start with purchasing expensive software and hardware assuming that the main part of the SOA initiative is accomplished. No, you can't buy SOA.

Someone has to do a feasibility study for your firm (preferably external vendor) – but this has to be an honest opinion of a qualified group of people. This study has to be convincing, it has to show if implementing SOA is the right choice at this time for your firm.

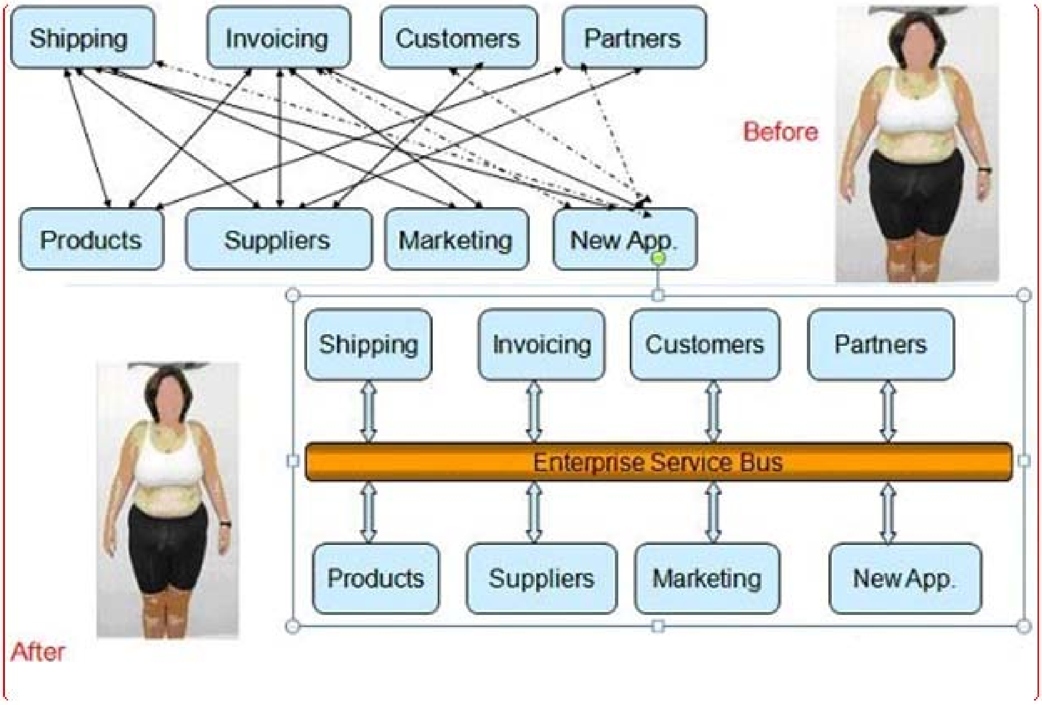
SOA maturity assessment is extremely important too.

Technical Benefits of SOA

If someone will create an inventory of all applications of a large organization and the data exchange between these applications, it may look scary.

An app A provides data to app B. It would be nice if app A would be able to easily send the same information to applications C and D as well, but this would require some data transformation. Currently, your people would just create and deploy a new ETL (extract-transform-load) processes for A2C and A2D data exchange.

But SOA can offer you a more elegant way by implementing an Enterprise Service Bus that would take care of the data routing and transformation. Someone said that the main reason for implementing SOA is reusability? Let’s look at the before-after SOA diagram:



The bottom diagram looks much better than the top one, but how much is it going to cost your firm to switch from the first diagram to a second one? How much would a reusable service cost you compared to just writing new ETL scripts for each new connected pair? Is it worth the money?

Why would reusable components would be expensive? Let’s talk about infrastructure, which is an enterprise level group that owns shareable software and hardware.

ESB Infrastructure

ESB (Enterprise Service Bus) requires a centralized supporting group with proper skills. How this group will be funded (a slice of each project’s budget, internal consulting, a hybrid)? SLA must be in a place that would require this group to accommodate every application the team needs in a timely fashion. The quality of service has to be clearly defined (is it 24x7 or what's the max time the service can be down).

Do not forget about service versioning, which is not the same as application versioning. Some service consumers may be happy with the version 3.2 and you'll have to support both 3.2 and 3.3.

Coming back to the human factor - do you have good relations with the ESB group? Personal relations will always get you farther than any SLAs combined. By the way, Joe Smith who runs the ESB group has his own career objectives and ambitions

The ESB group has to accommodate the application group's needs in a timely fashion. You need to make sure that Joe finds time in his busy schedule and allocate his resources to your project's needs.

If you keep your ugly point-to-point ETL way of connecting silo applications, you do not need to create this infrastructure and introduce yet another moving part into your rather complex enterprise architecture. Again, human relations between you and Joe Smith become crucial for the success of your SOA project.

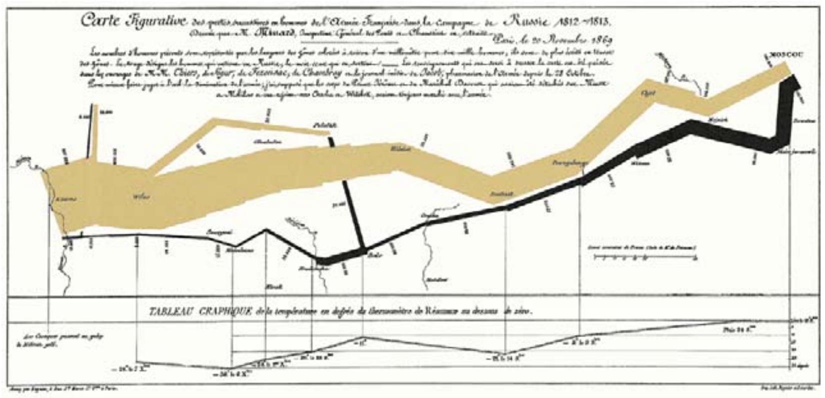
Now you need to get trained and spend some time preparing your data for input into the ESB in the proper format, specify the right output format for every new application and document it according to your firm's SDLC process. How quickly the infrastructure team will write and test the conversion script for your recipient application?

Yes, every new connection between the applications that you introduce adds a bit more spaghetti to the diagram, but you could have done all this with your own resources in a controlled timely manner without missing deadlines and jeopardizing your career. By the way, Joe Smith from infrastructure has his own career objectives. If your project fails, your yearly review suffers, now you need to find common ground in achieving yours and Joe's personal goals.

To SOA or not to SOA

This does not mean, at all, that I do not recommend implementing SOA in your enterprise, it just means that you need to be prepared and armed when it comes to dealing with all these issues. SOA is definitely a way to go in green field situations when you start automation of your enterprise from scratch.

I wonder if you are familiar with this diagram:



This is Minard's Diagram of Napoleon's March on Moscow of 1812. The width of the line shows the size of the army on the way to Moscow and back.

Napoleon was expecting greetings from Moscow authorities, but entered the abandoned and burnt city with no supplies and food for his huge army. His retreat was also a deadly enterprise because of an extremely cold winter (–36 ° F), no grass for horses (most of them were eaten by the French soldiers anyway). You can read more about it in Wikipedia.

My message to you is this: “Do not repeat Napoleon’s mistakes in your SOA endeavor. Do not start it unless you have the right expectations and your infrastructure will definitely be ready.

Making Business Users Happy

Oh man, I almost forgot about your business users! Will they happily greet you at the gates of their cities? How did you convince them that they should shell out a substantial chunk of money for implementing your SOA initiative? Do they actually know about this initiative or you managed to get funding because you play gulf with the CEO of your firm?

If this is the case, I have bad news for you – you can't avoid contacts with business user while immersing your enterprise into SOA waters.

Do they give a damn about what architecture you use to give them the data? Not really. They will definitely ask you a simple question, "OK, I'll give you X amount of dollars and allocate resources for a year for your SOA project. Will I run business differently a year from now? Will you provide me more analytical data in a more convenient form? Will my customer-facing sales force become more productive which will translate to more sales?"

SOA+RIA

To make end users more productive, we need to shift gears a little bit and recall that user experience really matters. Think of an iPod. How many people know the name of its competitor? What do you prefer – Apple's iPod or Zune from Microsoft?

Zune has a bit more functionality, but iPod looks nicer and the user interface is cleaner. InformationWeek has published an article listing 8 alternatives to iPod. Have you even heard of them?

Lots of people purchased a new Apple gadget called iPhone. $500? Not a problem. They want it now. Let me stress, it's still a phone with the ability to browse the Web and a modest disk space for storing music. From the services perspective is not a revolution, it's the User Experience that make it stand out and this is the main reason why people want it.

You may ask what all this has to do with SOA? People like nice looking gadgets, and they like nice looking program interfaces too! It's easier to sell SOA to your business users if the SOA client applications are convenient and slick. This is especially true for the customer facing applications. If your sales representative comes to a prospective client with a rich-looking application clearly showing your products, their job becomes a lot easier. If you offer your firm's financial analysts rich GUI interface with live chart and graphs to easily compare performance of two mutual funds, analysts will really appreciate it. You still need to program the services that will go and get the data, but having a great presentation layer is extremely important.

When you create or modernize your application, do not just think about powerful multi-processor servers, multithreading, and ESB. Start redesigning the users experience – it has to be available everywhere (at home, on the road and in the office), it has to be responsive, and it has to look better than client server application written ten years ago in Visual Basic and PowerBuilder.

Ten years ago we were thinking screens not servers. And this was not a bad idea. When Java was born, client programming became complex, besides, Sun Microsystems and IBM wanted to sell server licenses, which has moved most of the Java development to the server side.

Now we can talk about the rich Internet applications (RIA) that can and should be used as consumers to the services. The RIA term was coined by Macromedia back in 2002. I'm not sure if there is a formal definition of this term, but you can think of RIA as desktop-quality applications delivered over the Web with no (or close to no) installation required.

Let's turn on the time machine and go back 20 years. Mainframes with their dumb terminals dominates. The client portion of the application is a black screen displaying green and red text. Then, in the early nineties client-server application came about. The clients were not dumb anymore, rich looking GUI that utilized the processing power of the PC.

In mid 90th, Internet became popular. Plain looking pages had an ability to get all kind of information from a plethora of remote servers. From the GUI perspective, it was a pushback. Mainframe had black background with green letters, while Web pages had colored backgrounds, static pictures and the same text fields, buttons and rudimentary HTML tables.

Now the rich GUI is coming back again in the form of RIA. These are not page-by-page typical Web sites, but full-fledged applications with rich controls, audio and video, with state stored on the client. Now the user get this nice looking application delivered to their PC without the need to go through a complex install process.

There is a number of technologies that can be used for the creation of rich looking consumers of the services in your next SOA project. Leading enterprises started using professional designers for wire-framing next generation Web applications.

If you are an architect of an enterprise SOA project, do not just think servers, clusters, and grid computing. Think of your business users. Give them something nice, and they'll be helping you with your SOA-SHMOA efforts and will enjoy the new system as they already enjoy iPods and iPhones. SOA must improve the user experience of your business clients – do not underestimate the human factor.

Agility is a tough sell in enterprises

One day a group of smart software engineers from various companies locked themselves in the room and came out with great principles of developing software projects and published Agile Manifesto: http://www.agilemanifesto.org/ :

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

While I wholeheartedly agree with these principles, in large enterprises each of these principle is violated. As you know by now, the careers of the individuals are the main driving force for pretty much any project. Success of any IT initiative always depends on the following two aspects:

- technical principles and available tools
- motivation of the decision makers

Technical principles and tools

Ten years ago I was been programming using a fourth generation tool called PowerBuilder and my mentality was different: first, I was the best friend of business users, and second I did not really worry about what's under the hood. I could do stuff quickly, or using the modern jargon, I was an agile programmer without even knowing this.

I'd ask the business user Joe, "How do you usually run your business, what would you like to have on this screen, what step do you do after this particular one?" Sometimes, Joe could not give a clear answer, but I'd still give him a wide American smile: "No problem, I'll come back tomorrow and will show you something".

Mary, yes you, what's the most important word in my last sentence? No, Mary, not "I'll come back", but TOMORROW. Not next week, not next month, but tomorrow.

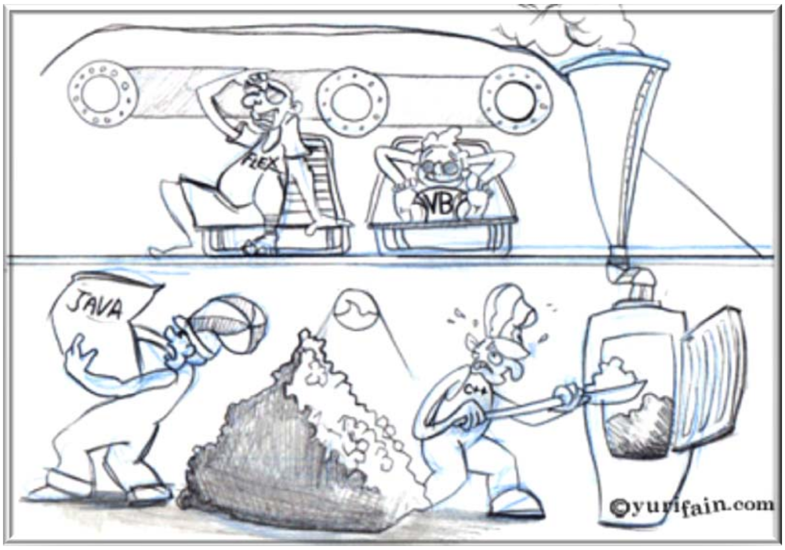
With PowerBuilder’s DataWindow and other controls it was easy. I did not have to pull Joe's teeth, I was able to create a working prototype in a day, show it to Joe next day, his glassy look all of a sudden would become friendly and understanding. Now Joe was back in control: "No, Yakov, you did this part wrong, I want it differently".

No problem, Joe, I'll see you tomorrow. Mary, what was the most important word in my last sentence? Good girl, Hasta mañana!

I did not really know how DataWindow worked, but I trusted this component. PowerBuilder used event-driven programming model, which was clean and simple. An object A triggers an event XYZ on object B, and this event can carry a payload - the data that the object B needs to operate. Using the modern jargon it's called Inversion of Control or Dependency Injection design pattern. Whatever. What's important is that the object B does not know about the object A. Loose coupling in action.

Then I became Java programmer, and my mentality has since changed big time. I realized that the user's screens are not that important, because I had an intimate knowledge of how programs worked internally. Screw users. I'll spend the majority of my time designing a multi-tier system that does not really depend on any specific screen and is universal. Joe kept still asking me, “when is our next meeting?” In a month. Mary, do not raise your hand. I see that you know the most important word here.

Why in a month? Because I could not do a decent working prototype sooner. We started to make fun of PowerBuilder or Visual Basic programmers who were thinking screens, while us, cool Java gurus, knew how the motor worked inside! These guys were enjoying a ride and counting cup holders, while we were thinking spark plugs and combustion chambers. We were enjoying the process of programming in and of itself.



A famous stand-up comedian Mickhail Zhvanetskiy from Odessa once said, "Who cares about the soup, when so much is going on in the kitchen!"

Now, with tools like Adobe Flex I started to care about the soup again, because I can. I can change the prototype twice a day, and Joe does the same with his business requirements. No six-freaking-sigmas documentation.

Napkin on the knee is back and it works

I'll give the final OK to my server side Java team only after Joe is 100% happy.

Besides, with Flex I can have the best of both worlds: the source code of the Flex framework is available, I can learn how it works inside and override it (not always it's as easy as it should be, but it's doable).

Working Flex promotes agile development.

Dear user, I'm your friend again! What do you want me to change?

CYA

If you are not familiar with this acronym, look it up at www.acronymfinder.com. Ok, if you are not online at the moment, I’ll tell you: CYA stands for Cover Your Ass.

For many project managers this is the main motivation in making any decisions.

James McGovern published a blog on How IT managers prevent hiring top talent. He names fear of losing power as one of the main reasons and suggests having less consultants and more people from the military.

I do not agree with these statements. These are some quick random thoughts on the subject:

1. Managers do not hire top talents because they can't afford to hire them.

2. Managers are not being afraid of losing the power, because top talent is not seeking power but rather an interesting environment and appreciation of his/her work.

3. If managers already work with a top talent (can be either an employee or a consultant), they often resists innovative ideas that come from these smart people. Why? Because the main goal of almost every manager is to move up the career ladder and not revolutionize software development. The innovative ideas of smart technical people put their career at risk. Yeah, the existing technology may not be of a bleeding edge, but everyone knows its features, and slowly but surely we'll get the project finished close to the set deadlines. This new technology sounds very interesting, but why take a risk being among the early adopters? Thanks, but no thanks.

4. For some reason, James does not like contractors that much. He does not want to admit that typically contractors are more innovative than employees. They need to keep their skills up to date and learn on the go, while some employees get too comfy in their cubicles over the years. I vote for inviting more contractors in on a short-time basis. I often work as a contractor myself. Currently I am working on three projects at the same time: two days a week on one, two on the other and Fridays on the third one. The managers who hired people from our company for short mentoring gigs made the right decision. My role is to mentor other people and make sure that the project goes in the right direction. Keeping me onsite for five days a week would be more expensive and unnecessary. I also prefer short assignments to long running projects.

5. Hiring people from the military may or may not work.

Pros: military people are often goal oriented, and if they decide to become good software developers/architects, they'll do it. Be all that you can be.. Recently I met a former marine in one the IT floors. He's a very respectful person, who does not have formal education in software but is one of the leading developers on a complex project.

Cons: military people may not have any talent in computers, and hiring them just because they are from military is wrong, in my opinion. Hire good software developers regardless of their past.

There is a very respected person, Carl Blum, whom I've known for years.. All his life he was recruiting software programmers and has turned this process into art. Now he's made the goal of his life helping people who are returning from the military service. I wish Carl success with this noble mission, but this is a little off the subject.

Part 3. Getting out of IT

And he was fired

I've got a call from a Java programmer I knew for years. He said, "I've got a problem. I was fired". Here's his story.

Joe was working for a mid-size financial company for several years. He knew Java and his application pretty well and felt safe there. On the other hand, since the Java job market was good, he started interviewing with some other companies, and was considering two offers. That day he left his USB drive at home and sent a piece of his code to his own personal Yahoo email account. He never received this email.

The next day, an HR person and his boss met with Joe and asked why he sent the email. Joe answered that he was planning to work from home, but never received this email. The boss said, "Please open your Yahoo email account now and delete this email if it's there". When he opened his email, they saw all his correspondence regarding new job offers and such… He was immediately suspended and next day received a letter stating that he was fired for an attempt to steal the proprietary software and the company may exercise their right to sue him in the future for breaching the non-disclosure agreement that Joe signed several years ago.

Most of us can work from home, and some of us have our employer's software installed on our laptops and home PCs. In my opinion, Joe is just plain stupid: he should not be sending this code using his employer's email. Every single e-mail in financial companies is being read by people from the information security department. It's not a secret, and has to be taken seriously.

I doubt that Joe's former employer will take any further actions, but Joe may have issues with his new employer. It all depends on how they check employment history. Joe may start working for a new company, and a month later get fired again if they find out about this story…

It's a sad story, but each of us should learn from it.

1. Do not use your employer’s email for sending anything that is not job related. If you are looking for a new job, use your personal email account.

2. Stop sending jokes or funny pictures to your colleagues. Some people do it every day just to let everyone know that they came early today… This also may get you in trouble unless you work for a mom-and-pop company.

Do not tell me, “’cause it hurts.”

After one of my Java talks, a woman from the audience came to me and said, "I'm being displaced. But that's okay; the company gave me enough time for retraining. I've been working with Java , but would you recommend that I learn .NET?"

This lady deserves a lot of respect for at least two reasons:

1. She has maintained her positive attitude toward her current employer.
2. She is ready to start learning again.

So last week you used to be a programmer, but not anymore. What's next?

Life After a Pink Slip

Do not panic. Start collecting unemployment (in the U.S. it's about $400 a week for most programmers). Incorporated contractors are also entitled to unemployment benefits - if you can't sign a new contract within three months, just close your corporation and get some cash flow from the unemployment office.

If you haven't found a new job and your unemployment benefits are ending in a month, find any school that is approved by the Department of Labor and register for their longest running course. Even if you won't learn anything useful there, your benefits may be extended for the time that you're in school.

Let me stress it again, your résumé should be customized for each job that you are applying to. The same facts about your work experience can be presented in multiple ways. As Nelson DeMille wrote in one of his novels, "He never lied. He could give 10 correct answers to the same question."

If, Else If, Else If

Even though this article is written for Java Developer’s Journal, I'd like to make it useful for non-Java programmers as well. Let's write some if-statements for a victim of a recent layoff.

```
if (urMainframeProgrammer){
/* Consider learning WebSphere application server, messaging, and
integration tools.

Get yourself IBM certified in these disciplines. For example,
WebSphereMQ administration is one of many career choices. Try
selling your industry experience. */

} else if (urVBorPBProgrammer){
/* Consider a database-related career. Improve your database
skills, obtain all possible certificates from one of the major
database vendors and present yourself in the job market as a
database developer who also knows a front-end tool. Visual Basic
people should master C# and ASP. Everyone should know XML. Learn
Silverlight and become an RIA developer. */

}else if (urJavaOrDotNetProgrammer){
/* Stay where you are, but make your résumé stand out from the
crowd by analyzing what the hottest skills are in your area using
job-related Web sites. Perform multiple searches using different
keywords that are close to your area of expertise, and
meticulously write down the number of ads for each keyword. After
finding the hottest skills, get yourself certified in this skill
by a recognizable software vendor like Sun Microsystems, IBM,
Microsoft, Oracle, etc. Do not send out your résumé until you've
received the certificates. Consider learning a tool for
developing rich Internet applications like Adobe Flex.*/
}
```

Moving Out?

Modern IT is a moving target and, if you can't keep up with it, you should find something more suitable for yourself. If you are considering opening a business (I hope you're not planning to get into real estate), be prepared to live without any income for at least a year. If you have saved some cash, this might be a good time to try to capitalize on that business idea you've dreamed about for years.

If you're not a businessman, stay in IT. Ask yourself this question: "How can I earn $60K a year in a non-IT world?" The most probable answer is to start by investing $50K+ into a business hoping to break even in a year and start earning some money. The other choice is to change your profession. Go back to college and in about two years you'll become a junior engineer or biochemist with a $40K salary and a similar amount of student loans.

Staying In!

Here's another plan:

1. Don't even start polluting the job market with your current résumé.

2. For the next six months sleep not more than six hours a day. Spend no more than three hours for food intake, wife/girlfriend, and kids. Spend the remaining 15 hours with books and your PC.

3. Find a couple of reputable courses that teach the programming tools of your choice. These weekly courses are expensive, but they give you a chance to listen and communicate with experts in this field. To make these courses more effective, take them after spending a substantial amount of time studying on your own. Install the trial version of the software, read the books, and study the source code of sample applications. This way you'll be better able to absorb the course info in the classroom and will ask the right questions in the class.

After taking the first course, continue studying at home and take another (more advanced) course in a month or two. Take any available free online courses on the subject of your choice. Attend professional seminars, user groups meetings, and sign up for each free technical Webinar. Check out the schools of continuing education at your local college. They may be offering evening/weekend classes.

4. Join an open source project (see sourceforge.net).

5. After all, of the above is done, add description of your new skills to your résumé and hit the job market.

Sorry for the cold-blooded coverage of this unpleasant topic, but victims of layoffs might already be sick of hearing that "It's going to get better any moment." Just don't stop fighting - looking for work is a full-time job and has to be done the right way. Keep pushing and they won't have any other choice but to hire you!

Come back - you can do it!

Thoughts of an aging programmer

During the last 25+ years I work as software engineer (the title does not really matter because most of the time I worked as a consultant). What's next?

Software Engineering is a very competitive profession. The question is if I can compete with a 30-years old software engineer from Bangalore? Should I leave the scene?

Roll over, Beethoven!

As of today, I do not have problems with employment charging several times more than most of the young offshore programmers. Will it last? Yes… for a while. I'm a down to Earth person and realize that if you lock me in a room with a 30-year old programmer and give us 30 minutes and an assignment to write a program that uses linked lists without using Google, I'll lose. They are faster. They type as typists... Fifty characters at the speed of sound, then 30 hits on Backspace...and then another 30 at the speed of sound...They know the names of the classes and methods in these linked lists, but they are not always sure when to use them. They pass technical interviews easily by studying the API.

Do I want to become a young programmer again? No. I've been there already. I've been programming at 25, at 35 and at 45. I'm better now. I'm wiser now and I'm happy to move forward, not backward.

I've been visiting my accountant who, and somehow the same question came up – do you want to be young again?
He answered, "Young - no, but I want to be 40 again".
"Why?"
"I just like the look and feeling of myself at 40."
"But you can work out in a gym and improve your look and feel."
"I know, but at 40 I did not need to work out…"

There was a period in my life when I started getting rejected by employers who were looking for Java developers. I was caught by surprise. Getting a job interview ALWAYS meant getting a job…I started working on my visibility by writing articles and book as I explained earlier in this book.

Now the situation is different. These days, I'm often being offered jobs without being interviewed. I did not update my resume in two years. People do not ask for it. I have a big mouth and just googling my name generates lots of materials (noise too) that often gives some managers enough reasons to hire me right away.

But once in a while I'm still getting these multi-person technical interrogations with poking needles under my nails. A couple of years ago, I went through two hours of interviews with a large financial firm. To my own surprise I still knew the answers to all the questions. And interviewers have not been shy.

This was a Java interview, but the guy asked me, "What would you do if you had to send a message using MQ Series, and you have a message in the ASCII encoding on one end and EBCDIC on another. How do you like this under-the-belt question? Anyway, I knew the answer, and said that since we're using JMS on the Java side, we can cast a generic *TopicConnectionFactory* to IBM's implementation and set a parameter (do not remember exact name) to specify that there is a non-JMS reader on the other end of the queue.

I knew this because I did it back in 2000. The interviewer exclaimed, "Did you guys not have MQ administrator? There is a configurable parameter that they could have set on the queue, so you would not even need to do it programmatically!" Then he revealed that he's working with MQ Series since version 1.0 (it's more than 10 years).

What can I say… I know, I did well on this interview, but I was rejected. The guy who sent me there simply said, "They decided to hire someone else". I can think of two reasons – either "my failure" with the MQ guy was crucial, or I just was too expensive compared to other candidates. No sweat. Today I have more projects on my plate than I can handle. Moving on…

Writing this book may not be to beneficial for my career, as it's not a politically correct book. I'm sure, some people will be angry with me after reading it. But on the other hand there's a lot of smart hiring managers who will appreciate it.

So why are employers still hiring me over the younger and less expensive candidates? One of the main reason is that they want to have insurance. If everything goes as planned, young programmers have no problems. Now raise your hand if your last five projects went as planned…

I've been working with well trained young programmers, who just panicked when they needed to provide a solution to a production problem in a high-pressure situation. Employers want to make sure that the project will move on if something unexpected happens down the road. They want insurance, because a failure of the project may hurt their career too. That's why they hire me, and I'll do my best to make sure they succeed.

This is THE ultimate goal of any seasoned consultant:

Make sure that the hiring manager succeeds.

The other factor that keeps me in demand is that I'm still learning new technologies a little earlier than others. I have a good nose for the next big thing in software and go there earlier than others. Two years ago, when all progressive mankind started moving in the AJAX direction for development of rich Internet applications, I started working with Adobe Flex. Today it's becoming hot, and I already have two years of experience under my belt.

Remember the golden rule?

Do the same things as others, but do them a little better than others.

I am not smarter than others, but I work a little more than others and adopt new technologies a little earlier than others.

It's 6 AM on Saturday. What do you usually do at this time? I know, I know…

I wrote this section of the book after reading a small and very smart book called “Tuesdays with Morrie”. My 12-year old son has read it by accident and said that adults can read this book too. Read this book, and then you might want to re-read this section again.

My friend is a 72-year old programmer

This is a short story about my friend Felix. The last 15 years prior to his retirement Felix spent working as a mainframe programmer for a large financial firm in New York City. He stopped working at 67, collected a well deserved retirement package and was looking forward to a new life traveling the world and meeting new people. His lovely wife Dora is a food critic who loves traveling and have been pretty much everywhere.. She runs a small travel agency “Travel Six Starts” 1-(212) 693-4444 for those who understand the difference between booking your travels online or via experienced and trustworthy travel agent. We often travel with Dora and Felix. We never feel any age difference because Dora and Felix are a lot more energetic and interesting people than many 40-year old people that I know.

To make a long story short, after a year of enjoying his retirement, Felix got a call from a former boss asking for help. Outsourcing of their system to young programmers did not work out, because the system was rather complex, and knowing the syntax of a programming language did not cut it – they needed people who understood ins and outs of this application. Felix signed a 6-month telecommuting contract paying very good hourly rate.

Needless to say, this 6-month contract turned into a 2-year gig. Finally Felix was done with this job and started traveling full time. We’d join Felix and Dora whenever our busy schedule permitted. Last year, we spent a week with them in Europe, returned back to the USA, while Felix,72 and Dora continued their skiing vacation in France. He loved this lifestyle and often expressed his happiness with the fact that his programming career was finally over.

But their travelling life did not last long - I got an email from Dora saying that Felix had to break his vacation and return to New York…to start a new 6-month contract with his former employer who was looking for him around the globe and managed to convince Felix to accept this offer.

It's yet another six-month gig, but let's not fool ourselves – he's facing yet another 2 years of programming.

I wish Felix all the best with his new contract. I know it's not about money but about being in demand, which is very important for any professional.

Many years ago The Beatles wrote a song "When I'm sixty four":

"I could be handy, mending a fuse
When your lights have gone.
You can knit a sweater by the fireside
Sunday mornings go for a ride,
Doing the garden, digging the weeds,
Who could ask for more.
Will you still need me, will you still feed me,
When I'm sixty-four."

Well, Felix asks for more at 72. Now I have a dream to get a programming gig when I'll be 72. What can be better?

Good luck, Felix! Many more contracts to come! Dora, do not be angry, let's plan our next skiing vacation. By the way, have I mentioned that Felix is a good skier too?

Conclusion

Several years ago my wife and I were vacationing in Thailand. We were sun bathing on one of the beaches in Phuket . A small yacht comes to the shore unloading tourists. The owner of the yacht is a European looking tanned guy with a ponytail. His business is to take tourists like us to a small almost desert island were beach is better, the water is cleaner, and they serve super fresh fish and crabs. This may be irrelevant for my story, but he lived in a bungalow by the ocean with a beautiful Thai girl who was twenty years younger than him.
This little yacht in Thailand was his way of retirement.

When I came back to the US, I told the story of this guy to Gregory Z., "What a nice way to retire! My wife and I have modest savings, which would be enough for both of us to move to one of the warmer countries and live there without worrying too much about project rollouts, functional specs, offshore developers… "

Gregory replied, "Maybe it's good for that guy, but YOU can't do this. You have to be involved."

He's right again. I need to be involved with IT - this is what I've done all my life, this is what I know how to do, and I hope that I'll stay involved with IT for as long as possible. Remember Hotel California by Eagles?

"You can check out any time you like, but you can never leave!"

To be continued…

Yours truly,
Yakov Fain