* Intro

(This talk was given at the UMass Boston CS Department
end-of-semester/alumni party, May 13th, 2014. These are my talk notes
-- I didn’t use presentation slides).

** Agenda

 - I’m not going to talk about the subtleties of C++2011.

 - Tonight, I’m going to talk about

   + college. I’m sure you’re sitting there thinking "but I know
     about college". But, I like to think of college as "14 of the
     best years of my life". So I’ll talk about college

   + life after college (aka "having a job")

   + going back to college

But not necessarily in that order

* The Music Career
** Music School

 - Not from Boston Area. Originally from Northeast PA.

 - Since age 12, wanted to be a professional musician

 - Came to Boston to attend Berklee College of Music. It was a great
   place to study music

 - Like most people, graduated with quite a bit of student loan debt.
   In my case $14,625. Doesn’t seem like a lot now, but college was
   cheaper then.

   Earned 9,300 my first year out of college. That debt was ˜ 1.5x
   the annual starting salary of a musician.

 - Most of the jokes you hear about broke musicians -- eating Ketchup
   soup, buying two cases of snickers bars and eating that for a
   month. They’re all true.

   Last day of last semester, emptied out all of the cereal containers
   in the cafeteria. That’s what I ate for a couple of weeks.

** Working in Radio

 - When you graduate from college, you expect to get a job.
   Preferably a job in your field. Doesn’t work like that for
   musicians. Steady work is very rare.

 - During my music career, played in bands, recorded albums, and did
   graveyard shifts at a WUMB, the public radio station on the UMB
   campus.

   BTW, WUMB is really a great radio station. If you haven’t listened
   to them, you should check them out. If you like what you hear,
   [public radio pitch voice] pick up the phone and dial 617-287-6999,
   call now, make your pledge, and become a member of WUMB.

   Yes, I used to do fundraiser pitches. Sometimes it was a lot of
   fun (Talk about early morning pitching with Ken on the Quiet
   Storm).

- I learned two important lessons from public radio fundraisers

  1. The importance of being able to sell yourself. Don’t be afraid to ask for what you want. Sometimes you’ll actually get it. Sometimes you’ll get less. But you’ll do better than not asking at all.

  2. The folks at NPR spend a lot of time studying fundraising. Takes five pledge breaks for the average listener to realize that you’re doing a fund drive.

     Remember this when you’re asking for a raise: five times.

* CS Career
** Heading back to school (I)

- Life in the music biz was very rewarding. Lots of opportunity to be creative, and work with different people.

- It was also kind of crazy. You work a ridiculous number of hours, you have an erratic sleep schedule, and you’re always broke.

  Kind of like being a student :)

- One day, I woke up, in the back of my car, and crawled out of my sleeping bag, for the third time that week, and said "you know, I don’t think I can do this for another ten years"

  That’s when I started to think about switching careers.

  This was 1997, when we were all on our dial up modems, and the web was becoming really popular. I was already doing a little web development in my spare time. Built a web site for the recording studio I worked at. I’d just discovered this really cool thing called NetBSD. Computers and tech seemed like a much better career path.

- I found out that UMB offered a CS certificate program. So I got in touch with Dennis Wortman, who ran the program at the time, and enrolled.

- At the same time, I emptied my bank account to pay off Berklee loans. Rationale: should pay off one education for starting another.

- I owe UMB a huge debt in this regard. UMB has a really good CS program, and it’s much more affordable than some of the other colleges around Boston. Going back to school really changed my life.

  All the time I’ve gone to UMB, I’ve taken two (and sometimes three) courses per semester. I was able to work, and pay as I went. Took a long time, but I was able to do it without going into debt.

** Getting First CS Job

- After enrolling in CS program, took all of the normal CS undergrad classes. CS110, CS210, CS240.

- Landed my first job while taking CS310. The way I managed to get it was by debugging GDB.

  CS310 was taught in C, and I spent a lot of time debugging C

```
   programs with GDB.  gdb has this really nice function named "call"
   which allows you to call C functions from within the debugger.
   Useful for calling "dumper functions"

   At one point during the semester, "call" stopped working; GDB
   immediately segfaulted.  Unix lab operators put me in touch with a
   fellow named Karl, who maintained the GNU software on UMB servers.

 - Karl: Yeah, it’s not supposed to do that.  He’d give me something
   to investigate.  I’d work on it, go back to him with what I found,
   and he’d give me something else to work.  I ended up spending a lot
   of time using gdb to debug ... gdb.

 - In the end, problem turned out to be the result of a Solaris
   upgrade: noexec_user_stack.  This was a new Solaris feature to
   prevent stack smashing.  Essentially, causes a SEGV when any
   program tries to move IP to a stack address.

   How does GDB "call" work?  Copies code into a new stack frame,
   executes it, then pops the stack.  To Solaris, looked just like
   stack smashing.

 - After several weeks of this, Karl said "You’re pretty good, and
   pretty persistent.  A friend of mine is looking to hire a few
   programmers.  Interested?"

   Programming job?  Hell, yeah!

 - Lesson: Hard work will help you be successful, but that’s only one
   piece of the puzzle.  At some point, you’ll need people who are
   willing to give you a break.  There’s an element of luck involved.

** Intuit

 - First software job: Intuit.  I worked for a group called "QBN" aka
   "Quickbooks network".  We wrote online services that were add-ons
   to Quickbooks.

 - First lesson about industry: sometimes professors will change
   homework requirements a couple of times, and you have to go back
   and re-do stuff.  This is practice for the real world.

   In the real world, most projects are always evolving, and
   requirements are always changing.  Some amount of requirements
   churn in normal.

 - Worked there as a contractor until I finished my major certificate,
   then was hired as an employee.

 - At that time, I was thinking "Finally, done with school".  Prof
   Simovici called me into his office; he had other ideas.  He
   encouraged me to pursue an MS.

   I had first job, had major certificate.  I was reluctant about
   punishing myself with another few years of grad school.

   Prof. Simovici said that having a masters degree would make me a
   much better computer scientist.  And also pointed out that Intuit
   was just a first job -- and I probably wouldn’t stay there forever.

 - Prof. Simovici was right.  And I owe him a lot of gratitude for
   urging me to pursue a graduate degree.

** Grad School
```

- Since I was an employee, I was eligible for Intuit’s tuition assistance program. If you go to school to improve the skills that you’ll use at work, the company will help you pay for it.

  Intuit payed for most of my masters degree. I worked part-time during semesters, and full time between semesters.

- Suggestion: if you’ve finished your undergrad, and are about to go looking for your first job, be sure to ask about tuition assistance programs. If you want to pursue a masters, this will really help.

- Doing graduate degrees + working can be tough. Eats up a lot of time. On the other hand, I always liked the having the ability to switch gears: When you get fed up with your job, you work on school projects. When you get fed up with school project, you go back to job-related things.

- If you’re considering a masters, I highly recommend it. Compilers and CS680 had a huge, and lasting effect on the way I viewed software development.

  Doing 681--683, I found there was a significant reinforcement between things I’d learned at work, and things I was learning at school. I was able to take some things from industry (build and test automation; project management) and incorporate them into school, and to take ideas from school and incorporate them into work (code metrics, cyclomatic complexity).

- Eventually, I was in the last semester of my Masters; Prof. Simovici called me into his office, and encouraged me to pursue a PhD. At the time, I’d been working and going to school mostly three semesters/year. Felt like I really needed a break. No, I was done.

  Prof. Simovici was right again. It would just take me another two years to realize it.

** Kayak (I)

- By this time, I’d been with Intuit for four years. Very engineering-focused based culture. Was working on a project called SPC - shared product components. Eventual goal was to write in an intermediate language that would be machine-translated into java and C#; java for server-side products, and C# for windows desktop products. Kind of a cool strategic technical initiative.

- My manager pulled me aside one day, to say he was planning to leave Intuit. He & some others (including another UMass alumni, Paul English), were planning to do a startup, a web-based travel search engine.

  BTW - we’d like to bring you with us.

  Of course, I said "no". Thought I had a really cool gig at Intuit, and the idea of screen scraping flight information didn’t seem that appealing.

  Plus, this is ˜ 2years after the dot-com bubble burst, so I was a little worried about that too.

- Paul is really into recruiting, and he happens to be extremely good at it. A while later, he called me again. I’d had a really lousy

```
   week at work.  I thought, well, sure why not.

   Kayak became a pretty successful company, so I was lucky that Paul
   was such a persistent recruiter.

** Big vs Small Companies

 - Going from Intuit => Kayak was a bit of culture shock.

   + Intuit is a publicly traded company, with thousand of employees.
     A lot of products are financial, they have public image and
     shareholders to think about.  It was a very process-heavy place.

   + Kayak started as a very small company (I was employee #9).  We
     were financed by VC money, plus investments from our founders.
     Process really boiled down to "do anything to help us generate
     revenue, and do it as quickly as possible."

 - Seems counter-intuitive, but both approaches are right; each
   applies to a different context.

   This is something to think about as you start looking for your
   first job.  Larger companies can be very structured and conservative.
   Startups are more adventurous, and you can find yourself constantly
   context switching.

   Mid-sided companies are somewhere in the middle.  Different
   companies require different process models.

 - Keep this in mind as you’re looking for a first job.  Pay less
   attention to brand name.  Ask lots of questions about day to day
   life -- what’s it really like to work here.  Because you’ll spend a
   lot of time in that day to day work life.

   "Move fast and break things" sounds kind of cool.  But there’s
   always someone who has to deal with the breakage.

   Dealing with breakage (aka, the operational aspects of working at a
   tech company) can provide endless opportunities for excitement.
   But it can also provide endless opportunities for frustration.
   Depends on your personality, and how the company does the balancing
   act.

** Big vs Small (II)

 - Kayak gave me an opportunity to learn new things.  Sometimes you’ll
   have to take on a project that you know nothing about, simply
   because someone thinks you’re the most likely person to figure it
   out.

 - Another difference between big and small companies: in a big
   company, you’re likely to be focused on one area of expertise.  In
   a small company, you’re likely to get your hands dirty in every
   area of the business.

   At Kayak, I started off doing product development and sysadmin,
   then moved into data warehousing and business intelligence.
   Overall, I probably spent more time working with the business
   groups.  I’m now reasonably proficient in finance and marketing
   speak, which is a good skill to have.

 - In small companies, you’ll learn more about making tradeoffs.  For
   example, sometimes the "right" tradeoff is to do a crappy
   implementation that’s just good enough to work -- and get it out
   the door quickly.  From there, you figure out whether it’s
   worthwhile to invest more work in your crappy implementation.

   Similar case: fixing ALL bugs in your software is not a cost
   effective thing to do.

   This can be a tough pill for a nerdy software engineer to swallow.
   It’s almost like you’re being asked to do bad work.  Keep the big
   picture in mind.

** Back to School Again

 - After I’d been at Kayak ~ 1.5 years, I was starting to feel a bit
   of startup burnout.  In process-oriented vs adventure-oriented
   spectrum, I like a balance somewhere in the middle.

   I think that all startups are a little crazy and dysfunctional;
   that’s just part of the startup environment.  I figured that the
   company would grow, and they’d have to address some of the
   dysfunctionalities.  But I also needed a diversion; I’d been
   through one bad case of career burnout, and wanted to avoid doing
   that again.

 - So, I decided to take a class.

 - It was CS734 Database internals.  It was one of those courses that
   I’d wanted to take, but never fit into my schedule.  It was being
   offered during the evening one semester, so I signed up.

   Turns out I really liked database internals.  By the end of the
   semester, I’d decided to enroll in the PhD program.

   As I said before, Prof. Simovici was right :)

 - At the time, Kayak didn’t have a tuition assistance program (they
   have one now; it began just as I was finishing my PhD).  Instead of
   going to school to get a job, I had a job so that I could go to
   school.

   Put another way: I decided to go for a PhD so that I could spend
   less time at work.

   I wanted to work on something cool that I was excited about.  I
   didn’t really expect to use any of the stuff I learned, I just
   wanted to learn it.

** PhD

 - Pursuing a PhD is a huge time investment.  Academically, it’s
   probably one of the most unstructured situations you’ll find
   yourself in.  You have to set your own pace, and figure out what
   your own end goals are.

 - One of the best pieces of advice I got came from Joan Bolker’s book
   "Writing your Dissertation in 15 minutes a day".

   The main ingredient in any PhD dissertation is glue.  You pour a
   bunch of glue on the chair, then sit down and write.

   Discipline to do this on a regular schedule is key.

   Having a support network is also key -- other PhD students to
   commiserate with.
```

- Picking an adviser that you can work with is also important. And I think I was very lucky to have Pat and Betty as PhD advisers.

- The most important ancillary skill I picked up: My ad-visor put a lot of emphasis on writing clearly. (More red ink than black ink). As a result, I had to put a lot of effort into improving my writing; that’s been beneficial to me in a lot of ways.

  If you want to work on your writing, I highly recommend "Style: towards clarity and grace" by Joseph Williams

** Kayak (II)

- I was at Kayak for nearly 10 years. During that time, it went from 12 employees to around 300, and we had offices in five different countries. We did three M&A’s before being acquired by the Priceline.

- My last couple of months there, I found myself falling into a rut: doing the same thing day after day. Constant 6 month backlog of work. I really didn’t see any opportunities for career development.

- So I decided it was time to leave.

  Something I learned in the Music biz. If you take a direction and find that it’s not working out, don’t be afraid to pull the plug and start over again.

- I don’t mean to speak badly of Kayak; Kayak was very good to me in a lot of ways. For some people, Kayak will be an awesome place to work. I was just no longer one of those people.

** Peddling My Resume

- After 13+ years of software development work, I was finally in the position where I had to peddle my resume around, and look for work. (Intuit and Kayak had come to me, but now I had to go looking around at different companies).

- Bright side: the job market for software developers and computer scientists is pretty good. Over the last few years, the recession has decimated the job market in some fields. But the tech sector has largely avoided that.

  At Kayak, I interviewed dozens and dozens of software developer applicants. Whatever we were looking for, there was always a shortage.

- Bright side: there are lots of companies doing really neat stuff. You just have to find the right one.

- Suggestion: as you start to go and interview, think about the kind of work that you’re interested in, and then find out what kind of companies are doing it.

  Everybody uses software, even if they’re not a software company. Any patent troll will tell you this. You can be creative with where you look.

- Also think about values. We like to think of software as tools, but those tools are going to be used for something.

  For example, I’m very much opposed to digital surveillance, and I spend a lot of my free time teaching people about encryption, security, and about strategies for digital countersurveillance.

Occasionally, I’m contacted by recruiters working for mobile app vendors, or web analytics companies. Generally, my answer is "no, not interested". I’ve done corporate surveillance for several years, and it’s something I’m no longer comfortable doing.

I’ll draw an analogy to the Manhattan project. Some of the scientists were excited to be doing really cool physics. Some were concerned about how an atomic bomb might be used, and refused to take part, or petitioned Truman not to use the bomb in warfare.

Of course, most of you won’t work on something so controversial as an atomic bomb, but you’ll still have to face the question "is this work something I’ll be comfortable doing".

- So, back to Peddling the resume and interview.

  Interviews really work two ways. You’re trying to sell yourself to company X, and company X is trying to sell themselves to you. They have every right to ask "why should we hire you", and you have every right to ask "why should I work here".

- I interviewed at 7--8 different places before settling on Ab Initio.

  Most of the interviews were pretty straightforward technical interviews. You get a problem, you ask a bunch of questions to round out your understanding of the problem, and then try to offer solutions, and point out the tradeoffs.

  After all, it’s always about tradeoffs.

- One question caught me off guard. Fellow interviewing me says "let’s cut right to the chase. Tell me about some of the things you’ve really screwed up".

  Question caught me off guard. Luckily, I had plenty to talk about :)

  In retrospect, I that was a great interview question. If you’re doing interesting things, you’re going to make some mistakes. You’ll have to clean up the mess you’ve made, and you’ll have to learn from what went wrong.

  Like I said, it’s a great question.

* Conclusion

- I’d like to wrap up on a positive note.

- I’ve gained a tremendous amount from my education at UMass Boston, and I think you’ve all made and excellent choice for which school to go to.

- Since you’re here, you’re probably set on a career in computer science or IT. That’s probably a good choice. The job market is good, and it tends to pay well.

- Don’t worry too much about your first (or second) job. Any job is just one step in a series of steps that you’ll make throughout your career.

```
   Odds are you’ll work for several companies throughout your career,
   and you might even change careers a couple of time.  There’s
   nothing wrong with that; we all change over time.



#  LocalWords:  Berklee WUMB NetBSD Wortman Solaris noexec SEGV QBN
#  LocalWords:  ons Simovici SPC UMass Startups dysfunctionalities
#  LocalWords:  Bolker’s Priceline countersurveillance Initio
```