2016-02-23

An Industry Guide to Becoming a Software Engineer

Bill Langenberg

Technical Manager, TripAdvisor

Tufts Alum ‘01, BS in CS

Topics

- Skills you need
  - What you should learn at Tufts (Core CS Skills)
  - What you may need to learn on your own (Software eng. in the wild)
- What employers look for
  - Building experience and your resume
  - Succeeding in a technical interview
- Final remarks and Q&A

Core Computer Science Skills

- Data structures
- Algorithm analysis
- Impact analysis
- Problem solving & debugging
- Coding fluency in at least one language

Core Computer Science Skills

DATA STRUCTURES

Know how common data structures work under the covers, and the performance characteristics (Big-O) of common operations like add, delete, and find.

- Hash structures (HashTable, Map, Set, Dictionary, Associative Array, etc.)
- Lists and arrays (Linked lists, ArrayList/Vector, Stack, Queue, Buffers, etc.)
- Trees and graphs (Binary, BST, Red-black, Suffix, K-D)
- Iterators
- Generics

Core Computer Science Skills

ALGORITHM ANALYSIS

Be able to read code and determine the Big-O complexity. This is often where that knowledge of data structures comes into play. This is important both for writing code and providing code reviews.

Coding patterns and data structure selection both play into the overall time complexity.

Core Computer Science Skills

IMPACT ANALYSIS (environmental empathy)

You want to enhance your app and track user behavior during the session to personalize the experience and make recommendations to the user. How can we do this in a scalable way?

Memcached (or similar)! Cool.

How much headroom do we need in the Memcached cluster so bad things don't happen? Umm…

Be able to estimate the byte allocation, at least by order of magnitude. This example relates to memory requirements, but DB connections/load, heap size, response time, and other performance factors are also important. Be aware of what your code is doing and how it impacts the production environment.

Engineers vs. Developers

In-depth knowledge of data structures and the ability to accurately perform algorithm and impact analysis is what separates engineers from developers*.

* In my opinion

Core Computer Science Skills

PROBLEM SOLVING & DEBUGGING

Have a methodical approach. In completely unfamiliar code, can you isolate a bug in a technology stack, and hone in on the problem? This is an essential skill.

CODING FLUENCY

Knowledge: Understand one language really well. Know the language mechanics and frequently used patterns and built-ins.

Speed: Once you know how you want to solve a problem, writing the code should be the easy task. You should be able to write code in your language of choice without an IDE.

Core Computer Science Skills

TYING IT TOGETHER

Writing good, maintainable code and being able to provide effective code reviews to your peers is critically important. Let's tie everything together with a code review. What's wrong with this simple Java function?

```
public static boolean isNamePresent(List<String> names, String searchName) {
  for (int i = 0; i < names.size(); i++) {
    String n = names.get(i);
    if (n == searchName) {
      return true;
    }
  }
  return false;
}
```

Core Computer Science Skills

TYING IT TOGETHER

My review:

```
// ! This is a public function. Add a short comment about what it's for.
public static boolean isNamePresent(List<String> names, String searchName) {
  // ! Missing null checks. Code defensively!
  for (int i = 0; i < names.size(); i++) {
    // ! List param is generic. This is O(n^2) if given a LinkedList is passed in
    String n = names.get(i);
    // ! Know your language: Strings in Java are Objects.
    // ! This checks that n and searchName are the same object, not that they're equivalent
    if (n == searchName) {
      return true;
    }
  }
  return false;
}
```

Core Computer Science Skills

TYING IT TOGETHER

A better version:

```
/**
 * Determines if a particular name is included in a List of Strings.
 */
public static boolean isNamePresent(final List<String> names, final String searchName) {
  if (names == null || searchName == null) {
    return false;
  }
  // Know your language: for-each in Java uses an iterator under the covers.
  // This is O(n) as expected, regardless of the List type supplied
  for (String n : names) {
    if (searchName.equals(n)) {
      return true;
    }
  }
  return false;
}
```

Software Engineering in the Wild

- Linux
- Choose the right tool for the job
  - Indispensable tools
- Basic security
- How to be productive

Software Engineering in the Wild

LINUX

Can you take a brand new server, install a flavor of Linux, and be able to get a basic webserver up and running? Can you manage code in a distributed environment?

- Get familiar with maintaining software installations with yum (CentOS, Redhat) or apt-get (Ubuntu).
- Set up a development and build environment
- Fire up an Apache webserver or similar (nodejs doesn't count; it does everything for you)
- Be able to use git and/or subversion
- Be comfortable going to the web for answers
- Understand RSA keys

Software Engineering in the Wild

CHOOSE THE RIGHT TOOL FOR THE JOB

How many lines of code would you need to write to figure out top 100 IP addresses hitting your website, along with the counts, from the access logs? The access log files are gzips (.gz), and there is one per hour for each webserver (you have hundreds).

```
[web520b] 93.31.208.167 - - [22/Feb/2016:12:25:28 -0500] "GET
/Restaurants HTTP/1.1" 200 25651 "https://www.tripadvisor.fr/"
"Mozilla/5.0 (Linux; Android 4.2.2; A1-830 Build/JDQ39)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.95 Safari/537.36"
"WEBTRENDS_ID=198.47.103.148" "HOST=www.tripadvisor.fr" PL=X OU=- P=--
1364443 1364157 UID=VstEiAoQIHIAANmM@DsAAAAW PR=https
```

Software Engineering in the Wild

CHOOSE THE RIGHT TOOL FOR THE JOB

How about just one line?

```
% zcat access_logs/*.gz | tr -s ' ' | cut -d' ' -f2 | sort | uniq -c |
sort -nr | head -n 100
 11350 205.139.141.54
  3592 70.42.224.10
  3418 10.16.202.212
  3411 10.16.202.211
  3273 127.0.0.1
  ...
```

Software Engineering in the Wild

CHOOSE THE RIGHT TOOL FOR THE JOB

We can get even fancier and super-charge it:

```
% find access_logs/*.gz | parallel --bar -j 4 zcat | tr -s ' ' |cut -d' '
-f2 | sort | uniq -c | sort -nr | head -n 100
```

This probably would have taken me hundreds of lines of Java to do in a nicely multi-threaded way, like above.

The moral of the story is to build workable skills in languages and technologies that are fundamentally different from your core language. It can save you a **ton** of time.

Software Engineering in the Wild

INDISPENSIBLE TOOLS

- C/C++, Java, C#, or another core, object-oriented language
- Python, or another script-y language for getting stuff done faster, with less boilerplate
- Bash shell scripting
- SQL (up to and including joins)

Software Engineering in the Wild

BASIC SECURITY

There are a few types of security vulnerabilities that you should be familiar with, both in terms of how to exploit them and how to harden your code against them.

- Compromises your systems
  - Buffer overflow
  - SQL injection
- Compromises your users
  - SQL injection
  - XSS (cross-site scripting)
  - CSRF (cross-site request forgery; AKA: man-in-the-middle attack)
  - etc… (JSONP, and more)

Also be aware of PII (personally identifiable information). Don't expose or log access tokens (e.g.: Facebook, Google), e-mail addresses, or other sensitive information. Don't even store these things in plain text.

Software Engineering in the Wild

HOW TO BE PRODUCTIVE: BALANCE SPEED vs. SELF-SUFFICIENCY

When you take a software engineering job, your employer is going to expect you to be productive. The best performing engineers are able to balance asking teammates for answers to get unblocked quickly against spending time to figure out the answer themselves.

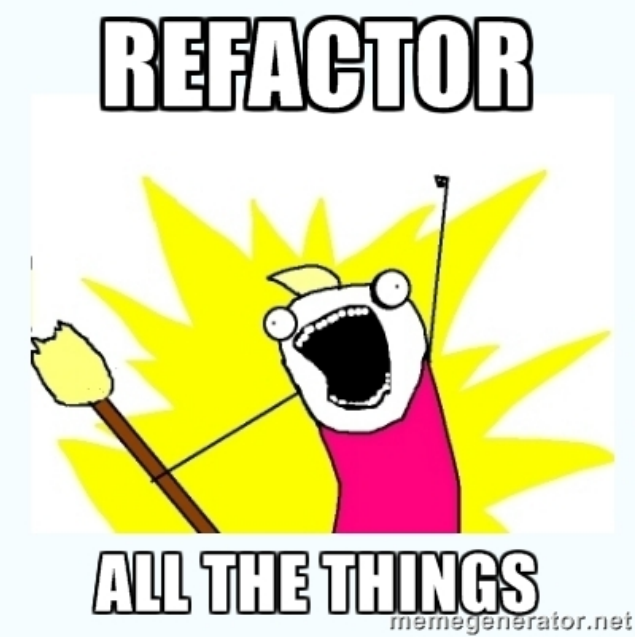
A general rule of thumb is to spend 30-60 minutes trying to figure something out yourself before interrupting someone else, unless it's a trivial question he or she can answer out of hand. Try to group your questions together and answer them in a contiguous block of time.

Software engineers need long blocks of uninterrupted time to be productive.

Software Engineering in the Wild

HOW TO BE PRODUCTIVE: BE PRAGMATIC

There's a very strong desire when you're getting started to refactor all the things. Perhaps it's because you find the code to understand, or doesn't match your world-view of good coding practices.



This is usually a bad idea. Refactoring takes longer than you think, and can have unintended and far-reaching consequences, particularly in a large codebase.

Leave the code you work on in a better state than you found it. Aim for incremental improvements. If refactoring is necessary, do so with measurable benefits (and do measure the benefit when you're done).

Software Engineering in the Wild

HOW DO YOU MASTER ALL THAT GREAT STUFF?

It takes awhile. Certainly longer than earning your degree. The old adage is true: practice makes perfect.

In addition to the CS curriculum the best thing you can do is experiment and keep coding. Participate in hackathons; build apps; find internships.

What Employers Look For

BUILDING EXPERIENCE AND YOUR RESUME

What I want to see:

- Quality internships in the field of Software Engineering and/or interesting personal projects that you *completed*
- An active coding project that you're excited about
- A reasonable GPA
- Completed courses that fill out the Core CS Skills from before
- Hackathon participation

When describing projects on your resume, I want a short sentence on what it's about, followed by what *you* specifically accomplished. It's important that I can separate your contributions from the team.

What Employers Look For

BUILDING EXPERIENCE AND YOUR RESUME

So, what if you don't get that summer internship? Jobs outside of software don't count for much. You're better off working on a coding project, or contributing to open source software.

I'll point you to Dan Blumenthal's excellent blog post on the subject:

http://dandreamsofcoding.com/2014/03/03/what-to-do-if-you-dont-get-a-summer-internship/

What Employers Look For

HOW TO SUCCEED AT THE TECHNICAL INTERVIEW

Nearly all top-tier software companies and start-ups have a challenging interview process. This is a good thing. The quality of your coworkers is represented by the rigor of the interview. You want to work with other talented software engineers, don't you?

Consider finding a job your job. Do yourself a favor, take it seriously, and PREPARE. Brush up on your core CS skills, and practice interview coding questions. You should be able to knock them out in 20-25 minutes, and be comfortable doing so without an IDE. Given technical interviews are largely the same, this will be time well spent.

Once you've been invited to interview, the most important contributing factors to getting a job offer are 1) technical skills, and 2) communication and "fit."

What Employers Look For

HOW TO SUCCEED AT THE TECHNICAL INTERVIEW

More on fit:

- Be able to talk about your experience and projects, both broadly and in technical detail
- Demonstrate you can accept constructive criticism well, and can you adapt your approach based on feedback. (Don't be stubborn or dogmatic)
- Exude enthusiasm about the company and the role you might have, and maintain a reasonable energy level
- Know a bit about the company you're interviewing at and their business? Spend some time exploring its site/app/product. Come prepared with thoughts on what you liked and what you might improve.
- Ask questions. What can you do to hit the ground running? What is the day-to-day life of an engineer like? How would you describe the culture. (Make sure you want to work there, too!)
- Know the company's dress code, and come to the interview just a notch above. (TripAdvisor is a casual company, so a tucked in shirt is fine. Please, don't wear a suit).

Final Remarks

The best advice I can give you is be strategic about your career. You may not know what you want to do, and that's OK. You can still have a strategy.

- Identify what you want to try, or perhaps which of your skills are lacking
- Do projects or find opportunities that will help you accomplish your goals
- Don't chase the highest paycheck, especially early in your career. Find the best opportunity.
- Periodically reflect and adjust

Q&A