Preparing for a Software Engineering Career

– Milos Curcic
  PhD in Mathematics 2012, University of Illinois at Urbana-Champaign
  Software Engineer at Bazaarvoice

My best general suggestion for landing a software engineering interview is to attend the UIUC Engineering Career Fair and Expo. Software companies like students with strong background in mathematics, so our math graduate students with some programming experience are likely to land interviews after speaking to representatives at these career fairs. It is a good idea to find out a bit more about the company and also to get a feel for what type of work the company does, what they are looking for and what kind of questions the students can expect during the interviews.

Furthermore, many recruiting companies are very eager to connect students with employers and will usually contact the applicants immediately after they submit their resumes on the recruiting company website. From my experience, these are rarely entry level positions and the interview process is more rigorous.

There are several excellent books on software engineering interviews that list some of the most common interview questions. I would highly recommend the following two books:

http://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/098478280X


There are several websites that review companies and describe their interview process and list interview questions - this is really helpful when talking to a specific company. Two of the main ones are:

www.glassdoor.com
www.indeed.com

I would suggest looking over these websites as they contain many brainteasers that companies like to ask and that are not necessarily programming questions. Going over these questions provides very good practice for the interviews.

Regarding the interviews themselves - I would suggest practicing a lot of interview questions as they do repeat, both technical and behavioral ones. The candidate will most likely be asked to write code on the board, so thinking out loud is important; the interviewer wants to see how he/she reasons about the problem. If the candidate cannot think of the best solution immediately, he/she should write down any solution (even if it is brute force) and then attempt to work out a better solution. If the requirements of the questions are not clear, the candidate should ask questions to get the clarification - the interviewers sometimes intentionally leave the questions vague and want the candidate to realize this and pose the questions as the additional requirements can lead to different approaches for the solution.
The behavioral questions often include describing a large programming project, previous team work and the difficulties faced during these projects, so a candidate should be prepared to talk about his/her projects.

Here is a list of main areas, courses or skills that are required for software engineering positions. I split them into mandatory skills, which are expected for a software engineer candidate, and optional areas/skills that can increase chances of landing the interview or job.

Mandatory:

- An introductory course in computer science - An applicant should have strong understanding of computer science fundamentals, such as binary operations, string manipulations, basic searching and sorting algorithms (and most topics covered in an introductory computer science course). Some of the most frequent interview questions are about character string manipulations, such as reversing a string, finding substrings, or counting characters in a string.

- A course in data structures - An applicant should have a good understanding of the basic data structures used in programming: arrays, linked lists, stacks, heaps, binary trees, hash tables, bitsets. A lot of interview problems reduce to using one of these data structures in a standard way. I have never seen an interview question which used any data structures other than the ones that I listed.

- A course in algorithms - Understanding “big O” notation is necessary for measuring the complexity of algorithms. Also, it is necessary to understand and to know the performance of the most common searching and sorting algorithms. A background in combinatorics, graph theory and probability really helps here. Also, knowing to apply the appropriate data structure can significantly increase algorithm performance. I would strongly recommend UIUC’s Algorithms course (CS 473), often taught by Jeff Erickson: [http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/](http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/)
I would also say that the following book is a must read for any software engineer, and definitely a great one to read in preparation for the interviews. In addition, the exercises often appear as interview questions: [http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262033844](http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262033844)
A good complement to this book, which is more of a reference and contains a large number of algorithms, is: [http://sist.sysu.edu.cn/~isslxm/DSA/textbook/Skiena-.TheAlgorithmDesignManual.pdf](http://sist.sysu.edu.cn/~isslxm/DSA/textbook/Skiena-.TheAlgorithmDesignManual.pdf)

- I would also say that proficiency in one production level object-oriented programming language is necessary. Knowing Mathematica, R, or Matlab is nice, but these programming languages are typically only used for prototyping and rarely for production level code. Knowing a language like Java, C++ or Python will look better on a resume. I would particularly suggest looking at the programming language Scala, which seems to be gaining a lot of popularity recently. Scala combines both functional programming and object oriented programming.
and for the more advanced topics the very popular F.A.Q. website:  
http://www.parashift.com/c++-faq/

For Java:  
http://www.mindview.net/Books/TIJ/

For Scala:  
http://shop.oreilly.com/product/9780596155964.do  
https://www.coursera.org/course/progfun  
https://www.coursera.org/course/reactive

- A solid understanding of object oriented programming fundamentals. An applicant should be familiar with object oriented concepts such as classes, encapsulation, abstraction, data hiding, inheritance, virtual inheritance. I would recommend the following source on the main concepts of object oriented programming:  
http://www.amazon.com/dp/0596008678/

- An example project: a candidate should show that s/he has some previous experience in programming a larger project and that he/she is truly interested in this field. Having a research project which involved some programming, participation in open source projects, online competitions (google code jam, https://projecteuler.net, kaggle.com) are all big bonuses. Ideally, a candidate would have an internship. Internships are much easier to land than permanent jobs and can really help with getting a full time job offer later.

Optional:  
- Object Oriented Design - a set of object oriented programming patterns which solves common programing problems. The following book is a must read for more experienced software engineers:  
http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612  
An easier read is:  
http://shop.oreilly.com/product/9780596007126.do

- Functional programming - a programming paradigm which is gaining more popularity recently. I think that the following book is the best computer science book and it covers the main aspects of functional programming:  
http://mitpress.mit.edu/sicp/

- Operating systems  
CS 423 - I recommend learning at least the basics of UNIX, learning how to use commands like grep, awk and sed for quick scripts in the UNIX shell. Also, the following book is a classic on operating systems concepts:  

- Numerical analysis / Numerical Linear Algebra  
CS 450, CS 458 - these courses explain how to perform some of the standard mathematical computations in practice on a computer.
- Web Application Programming - Software as service companies are on the rise and knowing how to program web applications is a huge plus. [https://www.udacity.com/course/cs253](https://www.udacity.com/course/cs253)

- Relational databases, SQL, NoSQL databases - CS 411

- Experience with threads, multi-core, multi-process programming

- Anything related to data science, data analysis, big data: [https://www.coursera.org/course/datasci](https://www.coursera.org/course/datasci)  
[https://www.coursera.org/course/dataanalysis](https://www.coursera.org/course/dataanalysis)  
[https://www.coursera.org/course/compdata](https://www.coursera.org/course/compdata)