William Harris Anderson
cell: 707-490-9942
email: williamharrisanderson@gmail.com

Objective:

Software engineer seeks fulfilling opportunity to work with computer software in active environment.

Goals:

Continue to develop skills in software engineering and programming. Create software applications. Solve interesting problems using databases, AI, and interactive graphics.

Education:

2014-2015: Attended NYU.

2010: Graduated UCSD University of California, San Diego with BS in Computer Science, GPA: 3.582. Transcripts available upon request.

2005: Graduated Analy High School.

Experience:

2013-14: Worked for the contracting company TalentBurst at Amazon's Lab126 as Android Applications Tester.

2012-13: Independent OSX developer working on audio tools.

2011: Software engineer at BrightScope (contract role for startup that lost funding)

2010 - 2011: Software engineer at Autonomy's Pleasanton offices (resigned – moved for personal reasons back to San Diego near UCSD)

2008 - 2010: Computer programmer for UCSD Attention and Perception Lab (part-time during college – 10/20 hrs per week during school, ~30 hrs/wk in summer)

2006 - 2007: Electronics assembly. Loaded printed circuit boards for Quaketronics (part-time during college - ~20hrs / wk, full-time in summer)

Skills:

Primary computer skills are networking, databases, analysis of algorithms and efficiency, data structures, Bayesian inference models, Markov inference chains, representing state spaces, and computer security. Languages most used are Perl, SQL, Java, Python, Bash, C, C++, and Objective-C/C++. Use of tools such as GitHub, Amazon AWS, Selenium, Jenkins, Beautiful Soup, XCode, Macromedia Flash, Fireworks.

Work Experience:

Working at Amazon’s Lab126 on the Amazon Fire TV, I heavily refactored and redesigned the test scripts used to repeatedly pair and unpair Bluetooth devices from the Fire TV. I designed a front end to the console testing script that added significantly more robust serial port support, error correction, data protection, analytical tools, and functionality to the various testing scripts. I also worked with programming and designing Arduino hardware to test Bluetooth devices.

Working on my own projects as an independent OSX developer, I encountered a problem to be solved. An important function was missing from the latest version of Apple's Quicktime software. Outside Apple software, eighty percent of the functionality existed inside command line tools such as SoX, and another twenty percent existed within C and C++ APIs for OSX. I chose to integrate this functionality into a UI that conformed to the OSX style. Taking example from open source software, I designed an application that placed a model-view-controller UI in Objective-C around C++ and open source C code, creating a pleasant and powerful user experience for those unfamiliar with command-line tools.

Working for Brightscope, my duties included fixing problems with the interaction of the website's front-end code base and the SQL databases, frequently repairing bugs and hotfixing production with the resulting patch. Also constructed and maintained a continuous integration server running Jenkins on the Amazon AWS cloud spread across multiple virtual machines. The tests analyzed the front end of the website and compared it to the SQL databases that were supposed to produce it. The continous integration machine also ran several scripts written by me in Python and Bash which monitored our GitHub repositories for pushes to arbitrary branches with names matching a regular expression, create a development version of the website, publish it with a cname related to that branch, and run a quick automated sanity check using Selenium and Beautiful Soup on the development version of the website.

Working for Autonomy, maintained and upgraded the SQL tables and ticket reporting features in their Digital Safe data escrow service. My work involved troubleshooting failed queries, inspecting and maintaining PostgreSQL tables, and creating and modifying shell scripts used for updating multiple production machines running in a cloud computing system automatically without causing downtime.

Working for Dr. Pashler in the UCSD Attention and Perception Lab, created a database that held data on experiments run by psychology professors, and a complicated front-end that represented that database in a graphical format, allowing them to quickly design schema and forms of their own for experiments, and then send the tests to experimenters, the results of which would automatically be saved to another database.

UCSD Projects:

Created software for a multiple-screen display and interface for Town and Country Learning Center, a San Diego Community Center for UCSD's Teams In Engineering Service program. The screen, created with ROCKS Clusters, consisted of nine synchronized monitors running on six seperate computers, which combined to display a single image displaying 3D applications run on a "head node" of a CentOS 5 Linux computer. Each machine in the cluster had to be synchronized across a network, and an additional development cluster had to be constructed of three machines and four screens. Changes made to the development cluster had to be pushed to the production cluster in the Town and Country Learning Center several miles away remotely. In addition to getting the display working, programmed several applications that required remote networking, such as combining Google Earth with a Dance Dance Revolution control mat to allow the children at the learning center to navigate the world. Also wrote a program to automatically display scientific videos downloaded from YouTube displayed in full-screen mode at scheduled times.

Programmed an IRC client in Java. This client had a graphical front-end interface, a number of user-configurable elements such as display color and size, a command history, multiple tabs open for each channel, and history files stored on disk of everything the user observed and wrote. This was designed to the specifications of a TA imitating a client, using agile design methods with myself leading a small group of coders.

Created a working compiler for Reduced-C, a scaled-down version of C. Starting from a provided lexer, created a one-pass parser to perform syntactic and semantic analysis of code. The compiler generates SPARC assembly language code. Handled recursion and subroutines in one pass using an arbitrary amount of memory on the heap and on the stack. Added features to overload functions, detect run-time array overflows, and null or dangling pointers into deallocated stack space.

Wrote a computer security application in Python used to test the strength of salted passwords by running a dictionary attack. The dictionary, a variable text file, would modify itself in real time, performing letter to number transpositions, lowercase letter to uppercase letter transpositions, and certain transpositions of letters inside passwords, as well as all combinations of these above elements. In a sample test file based on actual passwords, extracted 80% of all passwords in the first five minutes of running the program.

Wrote a Python Bayesian probability filter to find the probability of specific sentences based on unigrams and bigrams, using training data from a text file. This filter computes the log-likelihood of sentences based on the training data. This can be used to detect spam or other unauthorized messages.

Other school assignments included implementing a version of the NACHOS operating system, and implementing various data structures such as self-balancing binary trees, singly and doubly-linked lists, Bloom filters, heaps and treaps, and B-trees.

Created a Turing-complete 8-bit embedded microprocessor. Programmed in Verilog and Xilinx, the microprocessor was programmed to perform arbitrary modulus arithmetic operations on eight-bit numbers, detect the length of the longest string of 1s in an array of 8-bit numbers, and find the mode in another array of 8-bit numbers. Ran simulations and generated timing diagrams to prove the correctness and functionality.

Citizenship: USA