Tom Byrne
Robin Ambrozy
GBUS 420
Due: February 27, 2012

Table Of Contents:

Music Instructor, Musicland
Location: Amherstburg, Ontario, Canada
Dates: 1992-1996
Reported To: Mrs Farmer, Store Manager
Skills: Introductory Music Theory, Piano Instruction, Guitar Instruction

I worked part-time at Musicland during summers and after school from 1992-1996.

During this time, I was a guitar instructor for all levels from beginner to advanced. I scheduled students on a weekly and per-semester basis, based on our mutual availability and convenience. I taught guitar fingering, beginning theory, chords, transposition, ear training, sight reading, improvisation, scales, and rhythm structures.

I taught introductory piano and music theory as well; teaching triads, scales, modes, harmonizing, transposition, sight reading, basic rhythm and basic melodies.

Software Contractor, Essex County Board Of Education
Location: Essex, Ontario, Canada
Dates: 1995-1998
Reported To: John McLaughlin, Teacher on special assignment
Skills: Authority Software Package, C, Microsoft Foundation Classes, Borland C Builder, Photoshop, Windows 95

From 1995 until 1998, I worked part-time as a contractor for the Essex County School Board. The school board had an initiative to create computer-based activities that corresponded to the province's common curriculum. I was contracted to create the first versions of these activities.

The school board had licensed a software authoring package named 'Authority'. Using this package, I created several dozen Windows applications for grades 4 through 8. These applications covered various math concepts like multiplication and division. I also created applications for a variety of language arts programs that covered core concepts like sentence structure, word definitions and identifying nouns, verbs, adjectives and adverbs.

The next year, as my boss and I continued the program, we found that the Authority package did not allow us the flexibility we needed. We evaluated other development packages and chose Borland C Builder. Using C Builder from 1996-1998, we created more than a hundred additional independent activities and applications that are still in use today.

While creating these software applications, my boss would analyze the materials and give me sample activities from textbooks, workbooks or other sources. I would then design the application and Graphical User Interface (GUI) using Borland C Builder and Microsoft Foundation Classes (MFC) to be as close a match to the worksheet or source material as possible. This could be as simple as having the student type in an answer to a simple math problem, or as complicated as having them drag various shapes into a pattern.

Once we had developed a set of several dozen ‘activities’, we were able to rapidly create new ones by changing the data in them - for example, a simple worksheet of 10 multiplication problems could be easily made into 10 different worksheets, each with its own problems, by simply changing the data that was fed into the worksheet. This allowed us to cover a vast amount of curriculum with less work.

In January of 1999 I left this job, as I relocated to San Jose, California seeking full time work as a programmer.

Substitute Program Worker, Hope Rehabilitation
Location: San Jose, California
Dates: February 1999-May 1999
Skills: American Sign Language, Program Worksheets, Daily planning, Supervision.

I moved to San Jose, California in January 1999. While looking for a programming job, I took whatever part-time work I could find. I applied for a part-time temporary position at Hope Rehabilitation Services. I worked at Hope part time for 4 months, from February to May 1999. Hope Rehabilitation is a full time (5 days a week, 6 hours a day), community based day program for developmentally disabled adults.

As a temporary program worker, I would follow the instructions of the full time program staff. I assisted the clients in their day to day program activities. I followed clients' individual program plans(IPP's). Each client had their own set of goals and activities, which I would implement. During outings, I would assist them in purchasing their goods, and supervise them while teaching them fundamental living skills. I would assist the clients in preparing their lunch, which included ordering food and cooking during special events. I assisted severely disabled clients with their personal hygiene and toileting, when necessary.

I filled out daily reports for any client that I was responsible for, noting their attitude, activities, progress, lunch and personal hygiene. Several clients and one program worker were deaf. Working side by side with them, I picked up a basic knowledge of American Sign Language(ASL). The program had a licensed instructor from the local college come once a week to teach ASL. I attended these classes for two months.

I left this position when I obtained a full time position as a programmer at Petrasoft Research.

Software Engineer, Petrasoft Research
Location: Santa Clara, California
Dates: June 1999 - December 1999
Reported To: Gregory Stone, Owner.
Skills: Java, Swing, Ethernet & Telephony wiring, Windows 95 & NT Administration, Photoshop, Database administration (Sybase, Microsoft SQL Server), Microsoft IIS Administration, JDBC

I worked at Petrasoft Research from June 1999 until December 1999 as a Software Engineer. My boss, Gregory Stone, formed Petrasoft, a contract engineering firm in June of 1999 and I was the second employee.

When Petrasoft was formed, we moved into a small building that was unprepared for a tech company. I spent the first month preparing the office for our company to move in. I ran category 5 cable (CAT-5) to each of four offices, a server closet and to the basement where the telephone and data lines came in from the phone company. I had to wire each office with a data jack that ran to the server closet, as well as a phone line for each office. I hooked up all the wire in the server closet to an appropriate hub, and verified all the wiring and connectivity for data and telephone.

I was also responsible for setting up and administering a Windows NT 4.0 server for the office. I had to learn how to create a domain, add users, adjust user security, add and administer Microsoft Internet Information Services (ISS) and a database running Microsoft SQL Server for the 5 users that our office grew to.

I took a one week class in July: “Introduction to Java Swing”. I had done a little learning about Java, but not its Graphical User Interface (GUI) toolkit, called Swing. This class gave me a broad introduction to it.

Petrasoft took on a contract to re-write a Point Of Sale(POS) system for a local store. I was given the primary responsibility to write most of the application. This involved analyzing the old system, a 15 year old AS/400 with scanty documentation.

The first task was to get all the old data out into usable formats. With the help of our database programmer, we were able to dump all the old data to text files. From there, I had to design the new database tables and write code to normalize and clean the data. I was responsible for setting up a new server and database machine, installing Sybase AS 11, creating all the new tables, triggers and views for the new application. I then had to import all the data into the new database.

I then had to analyze all the workflows that the current AS/400 application was using. These were broken down into major groups of functionality: purchase orders, inventory management, data management, budgeting, sales management, reporting and administration.

Purchase orders required functionality to create, edit, delete and print purchase orders. Purchase orders are created to send to a vendor, but may be composed of products from multiple sources. Matching the data in the database to the correct vendors, with the correct unit prices, as well as matching the correct SKU numbers depending on the vendor and unit, required me to learn all about the procedures and processes for inventory management and ordering. I also had to write a printing framework so that the purchase orders could be printed out in the correct fashion for faxing to vendors.

Inventory management required functionality to track units that were present in the store or on order, to receive inventory when delivered (with or without a purchase order), or to manage the counts of inventory during periodic counting.

Data management required me to write UI to allow clerks to manage items in the database (such as adding a new item, or removing a discontinued one), manage prices on items (including sale prices, and the prices that were paid for the item from different vendors), as well as customer information.

Sales management required UI for listing the daily sales by department, and by register, so that the totals could be correlated with each cash register. At the end of every day, the sales from each register would have to be pulled to the central server, parsed and the data in the central database would have to be updated. This required a lot of database work. The inventories for each sold item would have to be updated, sales items would be created or incremented, and daily sales reports created.

Budgeting was a special request that was created for the new system, and didn't exist in the old system. The manager of the store wanted the ability to set a limit on the amount of purchasing that a department could do every quarter. This required me to write code that would look at all sales for that department for the last year, and calculated a weighted average for the current quarter. The manager could then raise or lower the budget for that department at will. The department managers were then able create purchase orders up to the limit of their department budget without managerial approval. This required me to create a rolling calculation every month when the monthly sales reports were calculated.

Reporting was required by the general manager of the store to be able to see all aspects of the store operations quickly. He needed reports to be able to view the amount of inventory, inventory on order, month to date sales, year to date sales, as well as breakdowns for sales and orders by employee and department.

The final part of the system was administrative. Managers needed the ability to create and delete clerks, update any clerk's purchase orders, change passwords and override purchasing limits.

This system was put into place in December of 1999, and I was on-site to support it as it went live January 1st, 2000. I worked to fix the last minute problems and integrate it into the existing cash registers and systems. I continued maintaining this software on a contract basis for a further 7 years (until 2007).

Petrasoft also took on a second contract, a piece of software named ‘Habanero’, which is a framework to allow companies to collaborate over the internet. A new company was formed and funded to develop this software, and the new company (Lightsocket, Inc) offered all of the Petrasoft employees positions, and the owner closed Petrasoft in January of 2000. I accepted the position offered with Lightsocket.

Software Engineer, Lightsocket, Inc
Location: San Jose, California
Dates: January 2000 - October 2000
Reported To: Gregory Stone, Vice President of Engineering
Skills: Java, Swing, Ethernet & Telephony Wiring, Windows 2000 Administration, Linux Administration (DNS, apache, users, source control systems)

I worked at Lightsocket from January 2000 until October of 2000. Lightsocket was formed and funded to develop a commercial version of software that was developed at the University of Illinois at Urbana-Champaign as a research project. This project was designed to be a framework for building collaborative tools over the internet. Lightsocket was designing, developing and marketing this software as a 'hosted platform' - programs that ran on our own computers.

I joined Lightsocket as the fifth employee and first engineer. I spent the first 5 weeks at Lightsocket doing physical wiring, running CAT-5 cable to each office and 12 cubicles, wiring data and phone jacks for each. I also researched server machines for our office, placed the orders for those machines, and set them up.

I was the system administrator for Lightsocket during the first few months. I was responsible for setting up a Windows 2000 domain server, and administering all the office users (which grew to over 30 before we hired a proper system administrator.) I also set up two Red Hat Linux servers to act as mail relays (using sendmail) and Domain Name Service(DNS) servers for our corporate intraweb (using Bind). I also installed and configured software packages to for source control systems(CVS), File Transfer Protocol(FTP) and Secure Shell(SSH) access.

I was also the primary UI programmer for our collaboration application. I was responsible for creating all the UI frameworks in Java & Swing for the collaborative applications to run in. I had to travel to Illinois to work with our remote team several times to integrate the server and client components together.

During this time, I was learning a lot about working within a large engineering team and the distribution of responsibility. Responsibilities were often overlapping and it fell to the engineers to figure out what each would be doing. In a larger or more mature company, a dedicated project manager or engineering manager would have that responsibility, but due to the startup nature of our company, everyone took a hand in these tasks.

We built a first generation of the framework with several collaborative applications, including a whiteboard, a movie player, a Virtual Reality Modeling Language (VRML) viewer, a messaging system and a collaborative web browser.

I took 3 weeks of vacation in August/September of 2000 to get married and honeymoon, and the week after I returned, Lightsocket was unable to get another round of funding to continue operations, and laid off 2/3rds of its workers (nearly every non-engineer), and 2 weeks later, closed its doors for good.

Senior Software Engineer/UI Architect, ePatterns
Location: Palo Alto, California
Dates: November 2000 - May 2001
Reported to: Curtis Cole, Director of Engineering
Skills: Java, Swing, UML, XML, Enterprise Java Beans (EJB), Orion J2EE Platform, Weblogic J2EE Platform, Object Oriented Design

In October of 2000, I interviewed for an open position at ePatterns, a small software startup in Palo Alto, and was offered a position as a Senior Software Engineer, which I accepted. EPatterns was a company formed in early 2000 by several Stanford graduate students and professors to do Complex Event Pattern processing.

My job was to join the UI team and help to create a User Interface to present the status of the system and display the events flowing through the system. Coming up to speed on the existing codebase and joining this team was very challenging. All members of the existing team had done postgraduate work in Computer Science, while I did not have the same vocabulary or theoretical background. I spent a lot of time reading CS texts online and learning a lot of theory that I hadn't been exposed to.

I learned a lot of design patterns and Object Oriented Design(OOD) - how to think abstractly about systems and programming. I also learned a lot about Application Programming Interface(API) design. Learning to design an interface that will be used by multiple systems is a delicate and precise practice, and I was part of a larger team that designed all the interfaces between different services, and between those services and the UI.

My primary task was to create a prototype UI for the first set of services that the server team had written. These servers took in a set of 'events' (an abstract term) and performed a set of analyses on those events. I created a prototype application that allowed the server team to view and modify their servers, as well as the objects and adapters in their servers.

In March of 2000, the management team was unable to secure a second round of funding, and in a cost-saving move, laid off more than half the company, keeping mostly key engineers. As part of this rearrangement, I was promoted from Senior Software Engineer to UI Architect, and was made the technical lead of the UI sections of the project.

We continued to develop an alpha version of our project, but the company ran out of money in May of 2000. The entire company was closed.

Senior Software Engineer, Netscreen Technologies
Location: Sunnyvale, California
Dates: May 2001- March 2004
Reported to: Les Arrow, Manager of Global Pro
Skills: Java, Swing, RMI, Dynamic UI Generation, TCP/IP, Firewall Administration, Object Oriented Design, Technical writing, OSI Model

I worked at Netscreen from May 2001 - March of 2004, in the Global Pro group. Netscreen is a manufacturer of firewalls, and the Global Pro group was developing a management platform to control many firewalls at once.

When I started, the group was small, 4 engineers and a manager, and my task was to learn a subset of the firewall functionality, and translate that into a set of user interfaces for the Global Pro application. I was tasked with learning how the firewall managed it's high-availability features, as well as it's DNS server, and creating UI to manage those features.

The application was broken into two parts, a UI application and a server application, and the two applications used Java's Remote Method Invocation(RMI) to communicate with each other. For every UI piece I created, I had to create the appropriate RMI calls and design the API with our server engineer to ensure that the correct data was passed through.

I also acted as the in-house Swing guru, and assisted other engineers with any difficulties, as well as creating and integrating a help system (JavaHelp) into the application.

In 2001, we released Global Pro 1.0, followed by 2.0 and 3.0 over the next two years. I was responsible for larger portions of the software as time went on, including writing specifications for other developers, and reviewing and refactoring other engineers' code for consistency and to ensure it adhered to our group's code standards.

In 2002, Netscreen acquired a company called OneSecure, and our group merged with OneSecure's UI group, and we started to work on a brand new project that was to be the next generation of our Network Management product.

In 2003, our group worked on developing a new product: Netscreen Security Manager(NSM). This product was designed to encompass all the functionality of Netscreen's previous product, Global Pro, as well as managing all OneSecure devices. My role in this new project was to be the technical lead for two major features. Netscreen Redundancy Protocol(NSRP) configuration, and DNS and Dynamic Host Configuration Protocol(DHCP) configuration.

This product, although built in Java, was entirely different than Global Pro. NSM was designed to have a dynamic UI, generated with a custom built framework. We designed a protocol for laying out UI elements and screens in a custom text format similar to JSON. These were called 'views'. For each view, we also defined a set of data mappings for the data, and the framework would map the data from the views down to the database. This is very similar to how current Object Relational Mappings(ORM) like Hibernate work.

I was responsible for defining the data structures and views for each of my features, and each feature involved 10-20 structures and screens. I also created and debugged problems with Swing components that were used by our framework to actually render the UI.

I spent the next few months finding and fixing bugs for our first Feature Pack(FP) release of our software.

In February of 2004, Netscreen was bought by Juniper Networks, and between February and March, we were working on integrating the two companies.

I was responsible for writing documentation for all features I had been responsible for, as well as analyzing and documenting code that had been worked on by engineers who had left the company. In April 2004 we formally joined Juniper.

Senior Software Engineer, Juniper Networks
Location: Sunnyvale, California
Dates: April 2004 - May 2005
Reported to: Jens Schmidt, UI Manager
Skills: Java, Swing, C, Packet Sniffing, UML, Firewall & Router administration, Object Oriented Design

I was a senior software engineer at Juniper networks from April 2004-May 2005. I was responsible for a large new feature in our project (now renamed Network and Security Manager) - Packet Inspection.

One type of internet appliance that Juniper produced was an Intrusion Detection and Prevention(IDP) device. This device would capture suspicious packets flowing through a network and save them to a special server. I was responsible for adding a feature to our NSM software that would retrieve these packets and display them in different fashions.

These packets came in a standard format: packet capture (pcap). Although a widely accepted standard, there is no java library for reading or parsing these packets. I was responsible for writing both a library for this, and for creating all the UI to display it.

During this process, I learned about the different layers of the OSI/TCP model, how to identify each layer of the packet (Internet Protocol(IP), Transmission Control Protocol(TCP), User Datagram Protocol(UDP)) and identifying several application protocols like Hypertext Transmission Protocol (HTTP) and File Transfer Protocol (FTP). Each packet would have to be read and parsed, and the valid information pulled out. This information would include source and destination Media Access Control (MAC) addresses, IP addresses, ports and application information, as well as the protocol being used (TCP or UDP). This list of packets would be displayed and the user could interact with them in varying ways, and sorted, based upon any of the content.

I was also responsible for maintaining my existing features, fixing bugs and expanding the data descriptions and views when new features were introduced into the firewalls.

In early 2004, I was approached by a colleague who was doing some contract writing for Syngress Press. They needed an additional person to act as a technical editor for an upcoming book: Configuring Netscreen Firewalls. I signed on as a contractor with Syngress Press to do technical editing and minor rewrites. Over the next few months, I acted as technical editor for 8 full chapters and wrote 5 additional sections that were under a page each. I was also asked to write the foreword when the previous author was unable to fulfill his commitment. The book was published December 1, 2004.

At this time, I wanted to move into a more senior role designing whole projects, so I interviewed internally for a promotion as a Software Architect. Since we were undergoing a management change, and a hiring decision for this position was put on hold until the change was complete. The delay extended past 3 months, and in the meantime, I was approached by Packeteer, who had an open architect position.

UI Architect, Packeteer
Location: Cupertino, California
Dates: May 2005 - May 2006
Reported To: Sylvia Lowden, UI Group Manager
Skills: Java, Swing, JAXB, Fedora Core Linux, Ant, Postgres DB

I worked at Packeteer from May 2005 until May 2006, in the User Interface group. Packeteer was a manufacturer of Internet appliances that performed Deep Inspection(DI) and Quality Of Service(QOS) bandwidth monitoring and shaping. The group I was hired into was designing and building a new product to replace the company's 8 year old management software.

I was hired into a group that consisted of a manager, two User Experience(UX) designers, and one junior engineer. I was given the position of UI Architect, and was expected to take all the designs and workflows created by the UX designers and code it.

The UX designers followed a process called Interaction Design, primarily created by Alan Cooper and Cooper Design. I attended a week long class in-house, that introduced many of the concepts that we were using. I spent the first few months working along side the UX designers doing paper prototyping, visual design, workflow design and UI design. We used several tools to document these, including OmniFocus and UML.

I worked directly with the server architect to design service API's and data objects for use when transmitting data between the server and the client. We used JAXB(Java Architecture for XML Building) to both generate our data classes, and to marshall and unmarshall data on each side. Our server platform was deployed on Fedora Core Linux, with our data being stored using the Hibernate ORM, and persisted in a Postgres DB. I managed several server devices, which included setting up Linux, installing and setting up Postgres, and managing and extending developer tools such as Ant and Apache Axis.

At this time, I was also designing and programming a UI framework for dynamic UI generation that would allow a programmer to ‘lay out’ a UI screen or ‘view’ in a simple XML format and render the views, while mapping each field to a data object. I implemented the entire framework and coded it in Java and Swing.

I was also given a job requisite to fill for a second engineer on my team. I worked with our technical recruiter to create and post an appropriate job description. I screened candidates and was the primary technical interviewer for candidates. I was the final decision maker on which candidate would be hired.

I was responsible for overseeing both junior programmers on a day to day basis. I documented and delegated tasks, tracked their progress, and mentored them both on our technical processes and on how to manage their time.

Our team produced an alpha product that tested very well amongst our users. Due to a senior management change, our group was disbanded and our project shelved. Instead of attempting to find a different position inside the company, I decided to leave, and was able to secure a position at Apple.

Senior Software Engineer, .Mac & MobileMe Groups, Apple Computers
Location: Cupertino, California
Dates: May 2006 - May 2008
Reported To: Jake Baumgarten, Back End Server Engineering Manager
Skills: Java, Swing, HTTP, WebDAV, Web Services, Multithreaded Programming, Apache Tomcat, XML

I worked at Apple in the .Mac Back end server engineering (BaSE) group from May 2006 until Mid 2007. The BaSE group was responsible for two major internet services: synchronization and storage. I was primarily working on the storage aspect, but was often 'on loan' to the synchronization team.

The storage group was responsible for a set of software running .Mac's "iDisk" service - a remote storage service that used the Web Distributed Authoring and Versioning(WebDAV) protocol to communicate with MacOS X and other clients.

I had very little experience with writing servers, so I spent some time learning the foundations that the servers had been built on, which was a modified version of Apache Tomcat. The protocols were WebDAV which is an extension of the standard Hypertext Transfer Protocol(HTTP). I read the specifications(RFC's) and developed a working knowledge of these systems and protocols that I could build on.

My first task was to decouple our storage from a disk-based system and make the persistence layer 'pluggable'. I created several backing store implementations for this, including the existing disk-based storage, a remote WebDAV backing store (that would write and read the data to any remote WebDAV server), and an in-memory based implementation for testing.

My second task was to work with our operations group to remove a set of proxy machines and replace them with a hardware load balancer. During this process, I was required to learn about our entire network architecture and routing infrastructure. Working with a small group, we broke our machines into separate banks and classified each bank for a specific type of internet traffic.

My next task was to re-write an in-house application that we used as a test harness. Our group used this application to run regression tests against our varying servers. This application was using outdated toolkits and was very difficult to modify. I rewrote the entire UI in Swing, and added a programmatic Command Line Interface(CLI). I worked with our build team to integrate it into our continuous build system, so that whenever changes were made in our servers, the appropriate test suites would automatically, and the results emailed to the engineering group.

I was then put in charge of finding a better solution for Apple's iWeb publishing feature. Apple's iWeb software was able to create and publish web sites to the .Mac web service, but the publishing solution was buggy and inconsistent. My boss and I designed a new publishing protocol. This protocol involved staging assets to a temporary server location and a special server operation that could walk existing published directory hierarchies and overlay new or modified resources. This set of operations allowed there to be a consistent view of the web assets. I wrote most of this code, which required a lot of refactoring and modifications to our existing code to deal with temporary locations, aging out files and resources, and cleaning them up periodically. I also designed and coded a background task processor that would handle these cleanup tasks.

My next job was to work with another server engineer to create the Application Service Interface(ASI) and write the code for .Mac Gallery - a new feature for publishing photos from iPhoto to the .Mac service. I wrote the ASI for publishing and retrieving the images and albums, as extensions to our WebDAV protocol. I also integrated a service for resizing photos on-the-fly, based on requests from the browser. These requests were run through a separate image processor that was Objective C based.

Throughout the middle and end of 2007 we began work on new features for our upcoming rebranding and release: MobileMe.

Senior Software Engineer, MobileMe Group, Apple Inc
Location: Cupertino, California
Dates: Late 2007 - April 2010
Reported To: Jake Baumgarten, Back End Server Engineering Manager
Skills: Java, HTTP, WebDAV, Web Services, Multithreaded Programming, Apache Tomcat, XML, REST

By the third quarter of 2007, I was fully working on Apple's rebranding of our web services, to be called MobileMe.

During the beginning stages, I was loaned to the synchronization team who were rewriting their entire service. I was given the task of creating a set of utilities to provide locking and atomic operations for their multithreaded server applications. The new sync servers were built on top of our storage platform, but needed these extensions. I resolved over a dozen high priority, critical issues to unblock their team and help them meet their deadlines.

My next job was to write new Application Service Interfaces(ASI's) for the upcoming MobileMe release. We were providing web interfaces to our iDisk and Gallery functions, and I was put in charge of exposing the protocols for these. We moved to using a new protocol, a Representational State Transfer(REST) based-protocol over HTTP.

The iDisk Web Service required a basic set of filesystem semantics - the ability to send, receive and edit files, as well as the locking and metadata semantics all needed to be exposed. I designed wrapper layers for exposing all these via a REST-ful web service. I also created new functionality to create archives (.zip files) on arbitrary collections of files, and exposed this functionality via a web service.

My next task was to create a web service for manipulating the .Mac gallery - adding, modifying and deleting photos and albums. This new RESTful protocol was much more powerful and terse than the previous WebDAV based protocol, and we petitioned to have all current clients (including iPhoto, iMovie and QuickTime) to transition to using it.

I wrote technical specifications describing the new Gallery Publishing Protocol and became our team's primary liaison to these client application teams. I would frequently work side by side with their engineers, looking at TCP traces and debugging the protocol or other difficulties during integration.

My next job was to work with another newer engineer on our team, and bring him up to speed on the Gallery Publishing protocol, code and server configurations. Once he had been trained, I turned over primary maintenance of this feature to him.

Part of our architecture for tracking assets in a user's gallery was stored on a set of disk appliances using BerkeleyDB. We had a heavily customized version of BerkeleyDB, and due to performance issues, I had to modify the BerkeleyDB source code to allow a multiple reader semaphore or single reader/writer mutex locking system. I worked closely with a database expert and made all these code changes, then worked with our Quality Assurance team to create a test matrix to catch any unexpected error conditions.

In early 2010 I was recruited to a project that would replace MobileMe.

Software Architect, iCloud, Apple Inc
Location: Cupertino, California
Dates: April 2010 - Present
Reported To: Patrick Gates, Director of Engineering, Jake Baumgarten, Cloud Services Manager
Skills: Java, HTTP, Multithreaded Programming, XML, Web Services, NoSQL Databases, Oracle, CardDAV, WebDAV, Python, Memcache, Google Protocol Buffers

I have worked in Apple's iCloud division from April 2010 to the present. When I joined the team, I reported directly to the Director of Engineering, and later was reorganized under a newly formed DAV (Distributed Authoring and Versioning) team.

As this team was brought together to create an entire new large-scale web service, I did a great deal of research. I spent several months researching libraries and frameworks for web containers, multithreaded and non-blocking input/output. Some of the technologies that I evaluated were Netty, Jetty, Mina, and Tomcat.

I did research into databases, persistence layers and NoSQL frameworks. I evaluated MongoDB, Cassandra, Oracle, Hibernate, Cages and Zookeeper.

I was put in charge of building an HTTP and base WebDAV server on top of a multithreaded and nonblocking I/O framework. This server later became the platform that iCloud would build its Personal Information Management(PIM) servers upon. This project involved doing programming work to store and retrieve both content and metadata for web resources. I converted an older integration test harness to exercise our new servers, and wrote and converted over one hundred tests to exercise base HTTP and WebDAV functionality.

I was also responsible for defining a new protocol for the storage, retrieval and synchronization of browser bookmarks across multiple devices. I worked with a client engineer and wrote a complete RFC defining this protocol.

I forked our base server and created the first implementation of a bookmark protocol server. I coded the alpha and beta versions of this server, and wrote the test cases. I also trained two other engineers on our protocol and code base.

I was then put in charge of a second server that had been built on our base, a CardDAV server, designed to implement RFC 2426. I inherited a half-complete codebase with an incomplete test suite, and was expected to turn it into a complete scalable web service.

I started by examining the test harness. The harness was written in Python, which I had little experience in, so I dug into it and started learning. Within a few weeks I had a decent knowledge of both the CardDAV protocol, and the test harness. I used this knowledge to expand the suite of tests to over 100 integration tests, and several dozen unit tests.

My next job was to write a set of tools to store binary data to an external source. I wrote a set of wrapper classes and utilities that allowed any persistent data store to be used, and switched at will.

Next I refactored all of the server code and removed places where we had duplicate code across our servers. I moved common code into our base server where all the other servers could benefit from it.

My next job was to re-write the lexical parser for our data. The RFC defines a specific set of requirements for handling vCard data, and we had been using an open-source library that was poorly written. I eliminated it and wrote a complete vCard parsing library as a separate project. The new library was over 500% faster and was less than 30% of the size of the original.

I spent time writing a performance test matrix to measure our capacity for individual users, individual servers, databases (connections and throughput), and for banks of servers. I worked closely with our performance team to create performance

test suites and debugging problems so we could evaluate changes to our system quickly.

During this time I learned a great deal about market pressures and evaluating tradeoffs in time and quality. I became adept at gathering data to make an informed decision about the impact of varying bugs or decisions. I learned that having data to back up a decision is absolutely necessary.

I worked closely with multiple groups and stakeholders while developing iCloud. I learned how to balance multiple requests and tight deadlines while under pressure. I was often the lone representative from our group while interacting with other application groups, web service groups, product management and marketing.

iCloud finally launched in October of 2011. Within the first 100 days, we had over 100 million customers. I spent the first few weeks coping with our launch and unexpected fallout. I then spent several weeks working with customer support to analyze and solve customer issues. I was always on call for high-profile cases, in the event that an Apple Executive escalated a difficult issue that was related to my area of responsibility.

I am currently still responsible for iCloud's contacts service. I am also currently doing research & development for future Apple products. I love my job, because I am constantly learning new things and am able to stretch myself intellectually. I also know that there are lots of opportunities for advancement.