Bank of America / Merrill Lynch
Summer Analyst Program

Glen Oakley

June 4 – Aug 10, 2012

Abstract

Bank of America is a multinational banking and financial services corporation, providing consumer, corporate, and investment operations for its clients. Bank of America strives to be a technology leader in the banking industry, and thus have a heavy focus on technological growth and development. I spent 10 weeks with the company's Global Wealth and Investment Technologies division, learning about the kinds of technology Bank of America leverages to secure itself as one of the largest companies in the United States. This article summarizes my experiences and knowledge gained from this opportunity.

Contents

1 Bank of America & Merrill Lynch

1.1 History

The history of Bank of America dates back to 1904, when it was knows as the California-based Bank of Italy. By the time it became known as "Bank of America" in 1930 through a series of consolidations, the organization was well on its way to become the largest banking institution in the country. From there, the bank started a national expansion that continues up to this day, quickly rising to the top through a series of mergers and acquisitions. One of the more recent and notable mergers occurred in late 2008; the purchasing of Merrill Lynch & Co., Inc. [1]

Merrill Lynch was also a long-standing financial company, getting its start in 1914. It relied on its strong brokerage divisions to bring success and profits. However, due largely in part to the 2007 subprime mortgage financial crisis, Merrill Lynch began posting heavy quarterly losses. For a number of strategic reasons beyond the scope of this article, Bank of America entered talks to purchase Merrill Lynch, which led to the purchase of the latter company for $38B, effectively saving it from bankruptcy. [4]

After the merger of these two powerhouse financial institutions, Bank of America began to rebrand all of its corporate and investment banking activities under the name "Bank of America Merrill Lynch". This has become the corporate and investment banking division of Bank of America. [2]

1.2 Structure

Bank of America prides itself on being a leader in technology for the banking and finance industries. While there is no official measure of this, the company has been implementing many features in all facets of business.

On the consumer side, the bank has been pushing features to the public that are intended to make the banking experience easier. An example of such a feature is the ability to deposit a check by taking a picture of it from a smartphone. The bank realizes that, in addition to making the banking experience easier for consumers, their technological prowess could also attract new, more technologically-savvy customers from the younger generation (who may not yet have loyalty to any one bank). I had little interaction with consumer business and technology, and thus cannot provide more detail on the consumer experience.

Internally, however, Bank of America Merrill Lynch is attempting to provide its financial associates with a vast array of tools designed to make the job of every employee in the investment and financial sectors excel at their jobs. In a way, for the technologists who work directly with software development, it's a sort of 'think tank' for business and financial technology. The Global Wealth and Investment Management Technologies (GWIM Tech, referred to here as "GWIM") division in particular is geared toward creating solid applications that can be used throughout the entire company. GWIM has created many applications that support a multitude of platforms. Some of these applications will be described later in this article.

The GWIM division of Bank of America Merrill Lynch is further subdivided into more specific teams and divisions. Each team focuses on a specific facet of development. The team I was assigned to is categorized under "Mobile Application Development". The Mobile Development division itself has hundreds of employees. My specific work was with the iPad Development team. There are relatively few people who work on this team (about five or six). The team's jobs involve working directly with iOS and its hardware; writing code for

the iPad and running test cases on devices are two examples of expected tasks for the team. The iPad team (as well as a few other teams) is managed by James Rajeshvincent, who was my hiring manager for the summer.

1.3 Culture

My work took place at the Hopewell Branch of Bank of America Merrill Lynch. This 450-acre complex of offices was originally held by Merrill Lynch as a major office space (1.7M sq. feet). Due to budgetary constraints, the campus is in the process of being sold off. [3] A few of the building are already clearing out, displacing many Bank of America Merrill Lynch employees. Similarly, while I was working there, a few of the buildings (not the one I occupied) were experiencing layoffs related to the downsizing of the campus. It was easy to tell that feelings amoung employees were mixed. In particular, employees in my building, including my team and manager, seemed unconcerned about the safety of jobs in the Mobile Development division.

My work took place in Building 1300 (one of about 10 buildings), on Floor 2. Most of the people I was around worked either on Mobile Development or on the Mainframe system for the bank. The mobile developers are divided up into different teams, including web services, backend development, UI design, and database management.

The team I was stationed with and interacted the most with was the iPad Development team. At the Hopewell site, the team mainly consists of four other employees (one of which who is actually a long-term consultant). There is, however, an off-site team stationed in India that is also considered part of the iPad team. In order to stay in sync, our two groups would have a private teleconference every weekday at 9:30AM (appx. 7:00PM India Standard Time).

I was mildly surprised by the diversity (or perhaps lack thereof) at the bank. My entire building consisted of people from all over the globe, including Americans, Taiwanese, Indians,

Latin Americans, and Europeans. In the microcosm of those I worked directly with, however, I found the area to be dominated by those of Indian nationality. In fact, my manager, three of the four people I worked directly with (the fourth was Taiwanese), and a large number of people on Mobile Development that I interacted with were all natives of India. At first, it was a bit intimidating being one of the few Caucasians that I interacted with on a daily basis, but this gave me the opportunity to learn a lot about the culture of India (which, unfortunately, includes a Monday-Friday 9/5 work week).

As Bank of America is a gigantic, multinational business, the company culture varies between divisions and locations. The experience I had may not reflect the diversity in other divisions. During my training, I was told that Bank of America has a very diverse culture, and the bank supports a number of cultural-oriented groups, i.e. a LGBT group and an Asian group. I could tell that Bank of America Merrill Lynch definitely cares about its employees, and works to make sure their environment is very conducive to creating a supportive and productive work environment.

2 "Briefcase"

When I arrived at Bank of America Merrill Lynch, the Mobile Development division was already planning a new application for the iPad, nicknamed "Briefcase". Because the proposal was so new, I was able to view much of the development process, which I have outlined in the following sections.

2.1 Mobile Development Platform

Bank of America Merrill Lynch has been utilizing mobile devices ever since the division's creation through the Bank of America / Merrill Lynch merger. These applications were written with the financial advisor in mind, and thus were mostly financial and business

applications. Applications were originally designed for Blackberry smart phones, as that was the predominant business device in use at the time. More recently, Apple's market share in the business sector has been increasing, due to a phenomenon referred to as "Bring Your Own Device" (BYOD).

BOYD replaces the old model of distributing computing hardware to employees. With the old model, employees would be given a laptop, phone, and/or PDA selected and approved by the company, and would be expected to always utilize those devices for work-related business. This made it easy for the company to support its own technology, as they had control over what their employees used. The recent trend in BYOD replaces this old model; BYOD allows employees to bring company-approved devices that they already own into their work environment, and use them as business devices as well as personal devices. Businesses like this new policy because it allows them to save money on high-end devices that would otherwise have to be purchased for employees. Employees also like this, as they can choose (to some extent) what device they want to use, and will most likely already be familiar with how to use their device. This also allows the company to utilize newer technologies, as employees are likely to purchase a newer device for themselves that they can then use at work. [5]

In response to the BYOD policy, the Mobile Development division set out to build their own platform on top of the platforms that already exist on mobile devices. The first devices to be targeted were iOS devices, in particular the iPad series of devices. The task of creating the platform itself was sent to the Mobile Framework team.

A large consideration when building the platform was security risks. In addition to standard company security issues, Bank of America has to meet government compliance requirements and laws that apply to financial and investment institutions. This includes documenting all transactions between employees and outside parties (an advisor and their clients, consultants, etc). A document or email leaking out of an application could have

disastrous consequences, so the platform was developed around the idea of 'containers'.

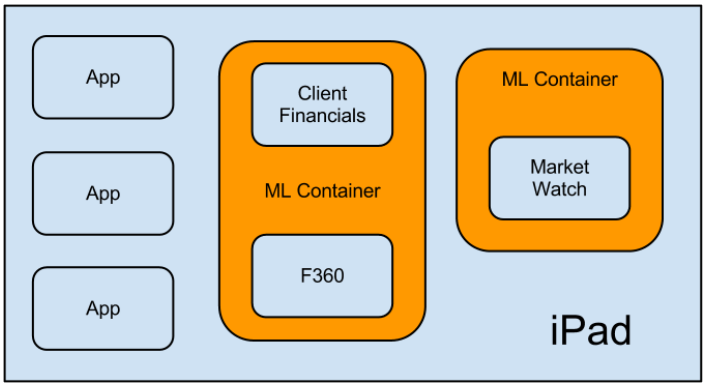

Figure 1: The theory behind containers running on an iPad

A container itself cannot do anything. It is not an application, but a sort of compartment for an application (or a few applications) to run in. Natively, iOS applications are installed to a device through an App Store, and can be launched from the home screen of the device. The Mobile Development platform changes the paradigm by making iOS treat a container as an app; the true applications lie inside the containers (as seen in Figure 1). This layer between a Bank of America Merrill Lynch application and the native iOS platform mitigates the risk of leaking or exposing data from the Mobile Development platform. This resembles the idea of sandboxing (segregating apps into their own environment), already used in iOS. Further security features of the platform (including network communications and file handling) are beyond what I was exposed to, and would probably get me in trouble anyway.

The Mobile Development team has created a number of apps since the introduction of the Mobile Development platform. These include:

"Good" App Used to interface with the Microsoft Exchange servers to give financial advisors access to their email.

Client Financials / F360 Gives financial advisors a full spread of data on every account for each of their clients.

Market Watch Provides consistently up-to-date information on markets around the globe.

Research Allows financial advisors to explore news and reports that may impact their clients' investments.

At the end of 2011, Bank of America Merrill Lynch took a survey of financial advisors that use these iPad applications. This survey had a twofold purpose. First, the Mobile Development division was able to determine overall feedback on these applications. The results were astoundingly positive, with especially positive results for the Client Financials application[1]. Second, the Mobile Development division wanted to gauge what kind of applications the financial advisors wanted to have available to themselves. The survey ended up collecting data about eight potential products (including already-released applications). Among these was a product that would allow advisors to access their business-related documents on the iPad as well as on the desktop. This idea evolved in to Briefcase.

2.2 Wealth Management Workstation

Originally, financial advisors were limited to working on desktop workstations. These workstations utilized a software suite developed internally at Merrill Lynch knows as the "Wealth Management Workstation". The WMW contains a plethora of applications used by the advisors, including those listed in Section 2.1. It also gives advisors the ability to move around their clients' assets and place orders with markets. The overall goal for the Mobile Development division is to make all the features of the WMW available on the iPad.

Amoung the desktop applications is a service called FileBox. FileBox is used by advisors to share documents with their clients. It is very similar to the popular public software known as Dropbox[2], but with limited capabilities and greater security. The Briefcase application is

[1] 100% of users gave overall positive feedback for Client Financials, and over 90% were recommending the application to other employees in different areas of the business.

[2] Found at: https://www.dropbox.com/about

essentially an extension of FileBox onto the iPad, with a few additional features.

2.3 Product Design

I was fortunate to come into the project early on. The design of the user interface had already been fleshed out, and the head of the iPad team had just finished writing 50 lines of introductory code; more of a testbed application than Briefcase itself. The structure of the application had yet to be determined, so it took about a week or two of experimenting and programming before the team had a design we were comfortable with.

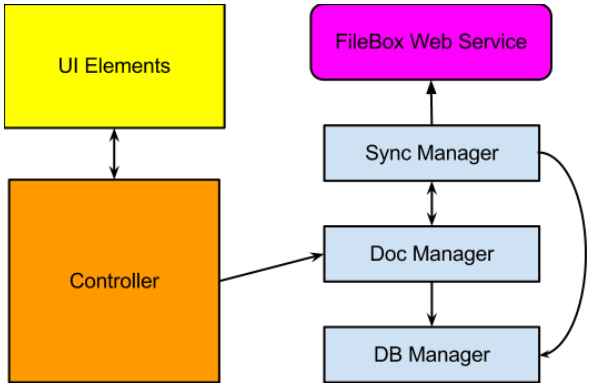

Figure 2: The design flow for Briefcase (views and controllers have been abstracted)

Briefcase's main purpose was to allow a financial advisor to sync important documents between their iPad and desktop devices. A user should have the ability to delete, view metadata (including date created, name, and client ID), and in the case of PDFs, view the document itself. All changes must be synced with the desktop, and thus the application would need to notify the FileBox servers of changes AND be able to receive changes made from the desktop[3]. Since PDF documents had to be displayable, the server needed to be able to provide the contents of documents as well as metadata.

Apple heavily encourages the use of the Model-User-Interface paradigm when program-

[3]We on the team forgot about the second part of this (receiving changes) until the last minute; it is currently not implemented.

ming iOS applications. This separates the user interface from the underlying data structures that make up the program, and allows them to interact with each other through controllers. We incorporated this directly into the design of Briefcase, as can be seen in Figure 2. The data was split into three different sections:

Document Manager The document manager's job is to track all of the documents stored in Briefcase (including their metadata). It provides the application's UI with the necessary document information and presents PDFs for viewing if needed.

Database Manager The database manager locally stores data associated with Briefcase documents. The local database ideally reflects the same data that the FileBox servers contain (plus additional local data). The database is especially useful for when Briefcase is unable to sync with the FileBox servers, as it allows the data to be retained and tracked until a connection is reestablished.

Sync Manager The sync manager is what interacts with the remote FileBox servers. The FileBox servers act as the central data store for the app, containing documents and their metadata. When syncing occurs, the sync manager sends information stored in the local database to the FileBox servers, and also updates the local managers from new data obtained from the FileBox servers.

The views for Briefcase were very simple. In fact, the entire application only consists of three views. The first view is a login screen. For the time being, there is only a placeholder login screen that does not do authentication against any sources. However, Bank of America utilizes many forms of encryption that will be incorporated into the application before it is released. The login associates Briefcase with a user account, allowing Briefcase to check for appropriate documents. The main view for Briefcase is a very basic table, listing a few select pieces of data to help the user identify each document. This list can be sorted through

tappable table headers. If the document is viewable (is a PDF), tapping its entry in the table will bring up a fully interactive view of the document through a native iOS PDF viewer[4].

2.4 Programming

Prior to the internship, I had no experience with programming for iOS, or even programming with Objective-C (the language most commonly used to write iOS applications) in general. The first week of the internship was spent going through tutorials and learning how to use the Objective-C language. My coworkers pointed me to the Stanford Online courses; the video lecture / slide combinations were very helpful in enforcing the new material. My coworkers were eager to answer any question I had, and pointed me to specific topics that arise often in the course of programming an application.

Historically, iOS Objective-C programming has made use of manual memory management. This is implemented through a series of methods on every object; 'retain', 'release', and 'autorelease'. Calling retain an object causes its retain count to increase by one, and calling release or autorelease causes the count to (eventually) decrease by one. When an object has a release count of zero, the object is freed from memory.

```
NSObject* object = [[NAObject alloc] init]; // alloc automatically retains
[object retain]; // retainCount == 2
[object release]; // retainCount == 1
[object autorelease]; // after the app completes a 'cycle', retainCount == 0
...
// object is freed from memory
```

Starting in iOS 5, the compiler takes advantage of a feature called Automatic Reference Counting (ARC). ARC allows the compiler to track the retention of objects without the

[4]For security reasons, the Mobile Framework team is looking into creating a proprietary document viewer.

need to manually declare releases and retains. It is essentially a form of garbage collection, but instead of the entire program getting swept for dereferenced memory, each object in memory is freed (nearly) the moment it is dereferenced. While it was introduced in iOS 5, ARC can be used for applications targeting iOS 4.3 and above.

Before Briefcase, the iPad team was writing all their code using manual memory management. As is custom in a large business environment, the team felt it too risky to update their code to take advantage of ARC (if it ain't broke, don't fix it). The Briefcase prototype was written with ARC, however, and those involved with writing the code felt that the new memory management feature was a great boon. With the release of iOS 6, Bank of America will no longer be supporting devices running anything earlier than iOS 4.3, meaning that the entire Mobile Development division will be able to take advantage of ARC for iOS.

Apple almost requires that developers make use of their native tools for building iOS applications. These tools are embedded in a program called "XCode", which is an integrated development environment specially configured for targeting Apple systems. In addition to providing programming and code features, XCode embeds layout and storyboarding features that allow a developer to visually design the user interface if they choose (as opposed to programmatically). This makes it easy to update the UI of a program and control the flow between different areas of an application.

Programming for iOS was a bit unlike any programming I had done before. The language makes use of messaging; 'messages' are passed between objects (through method calls), and many built-in objects have methods that allow themselves to handle data either synchronously (blocking) or asynchronously (non-blocking). As Objective-C is based off of the C language, every object in Objective-C is actually instantiated as a pointer to an object, which is then allocated and initialized; a programmer will almost always be dynamically allocating data to the heap. Unlike C, Objective-C abstracts object pointers to make them feel more like regular objects (in addition to ARC), which is helpful for those uncomfortable

for dealing with pointers.

I do not believe that my manager or team was prepared for how quickly I would pick up on iOS programming; the Briefcase prototype was completely finished within four or five weeks. Unfortunately, Mobile Development was re-prioritizing the applications being developed, and Briefcase was placed on the backburner for the time being. Still, the prototype met all of the functionality required for the first stage of the product, and with a few minor modifications, would be ready for testing and release.

3 ML Exchange (OSQA)

Since I finished my major project, Briefcase, much earlier than expected, my team and I started looking for work outside of specifically iPad development that I would be able to help with. Fortunately, Mobile Development was starting a new project nicknamed "Merrill Exchange".

In order to flesh out an idea for Merrill Exchange, many of us from different areas of Mobile Development got together and discussed an idea for a website that would aggregate knowledge from across the different development teams. The discussion quickly turned to Stack Exchange based sites[5]. It was decided that we would implement our own Bank of America Merrill Lynch version of Stack Exchange as a solution to the need for a knowledge base.

Before the next meeting, I had a chance to discuss my feelings with the one of the leaders of the project. I voiced my concern about the time it would take to build a Stack Exchange site (which was to have MUCH of the current Stack Exchange features, including live chat, user accounts, and a reward system), and proposed looking for a FLOSS (Free Libre Open Source Software) solution that we could modify to fit our needs. The idea was given some

[5]Stack Exchange is a network of individual communities, each dedicated to serving experts in a specific field. The sites contain libraries of high-quality questions and answers. http://stackexchange.com/

thought, and while everyone else worked on building a new site, I was assigned to track down a FLOSS solution compatible with our platform and goals.

3.1 OSQA

OSQA (literally Open Source Q&A) is a Q&A site whose original design was based directly from Stack Exchange. The software contains many features, including accounts, tags, voting, 'badges', and search[6]. The look and layout is very similar to Stack Exchange sites, which makes it easy for users to migrate over and intuitive for new users to learn. This one one of a few FLOSS Stack Exchange implementations I found, but this was the only one that was met with approval by the project leaders.

OSQA runs on Django, which is a web framework for the Python programming language. Bank of America Merrill Lynch does not use Python except for coding small tools, so no one on the team had much experience with the language. My own knowledge of Python was limited to self-contained, linearly-executing scripts of no real value. This would be the second language I was largely exposed to during my time at Bank of America Merrill Lynch. Many teammates were skeptical of the capabilities of Django and OSQA, so they continued to develop the made-from-scratch version of Merrill Exchange, while I worked on setting up OSQA in a virtual machine to prove its effectiveness.

3.2 Authentication

After just a few days, I was able to have OSQA up and running, completely with Merrill Lynch branding and test questions. The only hurdle left was authentication. OSQA supports a large range of authentication methods, including OpenID, local account creation, and Lightweight Directory Access Protocol (LDAP). It was determined that no one would want

[6]Search was a very lofty goal for our design-from-scratch version of the site, almost considered unachievable.

to use Merrill Exchange if they had to create an account. Since the site was to be hosted within the Bank of America Merrill Lynch network, it would be easy to authenticate against the company's account servers. Thus, I started exploring authentication with LDAP.

LDAP is a protocol for storing and accessing a set of records for a number of users. An entry in LDAP consists of some number of attributes, and each attribute has a name and value(s).

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
```

In the above example, one can see that a number of entries (distinguished name (dn), given-Name, surname (sn), etc.) are associated with attributes (cn (relative distinguished name) is associated with 'John Doe', and dn is associated with a cn and two dn). Bank of America Merrill Lynch already has many LDAP servers set up across many domains, against which we were able to authenticate. However, there was a non-technical problem with this authentication. In setting up LDAP, the company had chosen... 'less than desirable' names for the LDAP entries. Many entry names did not match up with standard LDAP Data Interchange Format field names, and some entries had completely irrelevant names (e.g., the email address entry was named something to the effect of 'entry_(1)'). Nobody (at least nobody we came into contact with) in Mobile Development knew how to interpret the LDAP entries, and we were unable to decode enough of them to create a reliable login and account system for OSQA.

The backup plan to LDAP that we ended up having to implement was to use HTTP POST methods. This way of authenticating was rather undesirable, but was all that was

available to us at the time. What would happen is the user would be taken to a site, where they would login with their standard Bank of America ID and password. This site would redirect them to the Merrill Exchange OSQA site; the redirect would contain the information needed by the site (name, email, etc.) in the body of the HTTP message (JSON encoded). This authentication method did work for us, and is currently what the site uses to authenticate users. When I left at the end of my 10 weeks, the site was nearly complete; a few team members were still working on a chat page, and the site was still hosted on a private virtual machine.

The OSQA site gave me a solid amount of experience with web-based protocols and ideas. I learned about HTTP GET and POST methods, types of data encoding, LDAP structure, and how servers handle web requests and return data to a requester. I have been planning to look more into the HTTP protocol and see what capabilities it has, and have grown to love python for prototyping programs and writing smaller pieces of software.

4 UI Automation

With two or three weeks left to go, I again had little work to do. Briefcase was still in limbo, and Merrill Exchange had little work left to be done on it. At this point, I had been playing with JavaScript in my spare time. As I was on the iPad team, my manager wanted me to continue work on the iOS platform, but there was little for me to do in such a short time frame. Fortunately, I learned that someone on another team had been experimenting with automating quality assurance testing on the iPad through JavaScript. This was the perfect opportunity for me to use my newly-learned JavaScript skills to help reinforce what I had done on the iPad.

Apple provides a framework for the automation of UI testing. It is one of many tools (called 'Instruments') contained within XCode that allows developers to examine their appli-

cations leaks, bugs, and resource usage. UI Automation is used to run the program without the need for user interaction. UI scripts can push buttons, move sliders, fill out forms, swipe the device screen, and even close an application. The output from the program ranges from success/fail messages, to a top-down list of all the UI elements for a particular view of the application. It takes advantage of the accessibility options associated with UI elements (the name of a navigation bar, for instance). With just a few small scripts, a developer can create a UI Automation test that visits every view of their application, checks for proper behavior, then exits with a log of every error that might have occurred.

The UI Automation tool in particular isn't well-advertised, in large part due to the lack of documentation provided by Apple[7]. It took a few days for me to really get started working on the automation; online tutorials were incomplete or uninformative for someone just starting out, save for a few hard-to-find guides. Once I started writing my own UI scripts, things started to feel familiar. JavaScript is uncompiled and runs in a linear fashion, making it easy to follow and change the path of the script as it iterates through the user interface.

I had multiple interactions with coworkers who weren't afraid to tell me exactly how they felt about the quality of feedback they'd receive from Quality Assurance tests, so the need for automated tests was twofold in that it would decrease the heavy workload on QA testers, and also provide more detailed, programmaticfeedback for developers. The hardest part about UI Automation at Bank of America Merrill Lynch was the lack of coordination between the application developers and those working on UI Automation. If UI scripts cannot determine the 'name' or identity of a UI element, they have to determine it from it's position on screen (represented by an array or nested array). If the layout of the screen changes (say a button is added), this can completely break the script. The developers were not adding most of the accessibility features to the UI design, so the UI scripts are very brittle and may break if a

[7]Apple provides no high-level documentation on UI Automation; the only documentation they have is a detailed list of the JavaScript classes used by UI Automation.

feature is added/removed. Hopefully, if the UI Automation way of testing takes hold, the iPad developers will concentrate more on adding these additional UI features.

5 Community

Over my 10 weeks, I had the opportunity to be involved in a number of events at and related to Bank of America Merrill Lynch. The internship coordinators were very active in getting the interns involved in a multitude of programs, and my hiring division even held an event or two.

5.1 Summer Intern Programs

The Summer Analyst Program coordianters had organized a few events to give us interns an idea of what Bank of America Merrill Lynch was like on a professional and personal level. Every couple weeks, all the interns from around the country were invited to attend live events (usually through webcasts) given by high-ranking members of the company, who would speak about different topics such as diversity, communication, and motivation. One of these presentations was split between the New York offices and the Hopewell branch at which I worked. We were given the opportunity to talk with senior managers and ask them about their rolls in the company and how they view Bank of America's future. It was a wonderful experience to learn from those who have been with the company for a long time and know the culture. I was also given the opportunity to meet with the hiring managers during one of these events. They were eager to share about what kinds of schools they target and what they tend to look for when recruiting. I was also informed that the internship program is considered very competitive, something I had heard before, but hadn't really had a chance to notice before meeting with the other interns at these events.

In addition to the official, on-campus events, the internship coordinators had set up a

few fun after-work networking events for us at the Hopewell branch. These were events such as bowling, going to a baseball game, and community service. The intern group this year was very diverse, and there were people working not just in technology, but in business and finance. At once such event, the new hires were also invited to discuss what it is like working full time at the company. Having the opportunity to discuss what kinds of work each of us did gave me a better view of the what kind of work makes Bank of America Merrill Lynch the powerhouse that it is today. Every intern I talked to had drive and passion (to some degree) for their work. At this point, it was easy to see why the internship program has been called 'competitive'.

5.2 Lunch 'n Learn

Halfway through my time at the company, Mobile Development decided to give an informative presentation during lunch in one of the large conference rooms above the main dining hall. The idea was to give all the employees on campus a chance to see what Mobile Development was working on, and to hopefully get them excited about past performance and things to come. My team gave a presentation about the iPad apps that have been developed, and what was in the works (including Briefcase). It was interesting to see the variety of people who showed up to the presentation. There were many financial advisors, who were very satisfied with the current selection of iPad apps and eager to see what would be coming down the pipe next. There were also developers from other teams, such as the mainframe group, who were interested in mobile development and wanted to get a taste of what the mobile teams worked on. The presentation helped me see the scope of the Mobile Development division.

5.3 Internship Presentation

As a requirement for the internship program, I had to give a presentation to a select group of people at the company, demonstrating what I had worked on during the summer. This was a great opportunity for me to start preparing for this paper and college presentation. I was a bit surprised at how disorganized the process of presenting was. Every intern was supposed to give a 30-minute presentation, but they all had to be scheduled within the same week. This led to confusion over whether or not the internship coordinators would be able to attend everyone's session, and in the end, some people, including myself, only presented to their manager and those on their team. I was surprised that the hiring manager had not anticipated this, and they handled the situation poorly. My presentation was well received by those who attended and helped remind me of some of the details of what I had done thus far.

6 Conclusion

Overall, my work at Bank of America Merrill Lynch was a very fulfilling experience. I was exposed to a plethora of new technologies that I most likely would have never experienced if it weren't for my time at the company. The people I met gave me the opportunity to form invaluable new professional relationships.

That being said, I do not think I would go back to work for Bank of America. I was told by many people time and again that the company has a spread of technologies, and that there are opportunities for everyone. I do not believe, however, that the kind of work I want to do is something that Bank of America involves themselves in. The career I wish to follow involves working close to hardware, and this is not an experience Bank of America can properly provide for someone with such interests. On top of my interests, I am not sure I enjoy working in such a large institution. I was blessed with working on a small team, but

I was always reminded of just how many people work for a company like Bank of America. It made me almost feel less satisfied with my work, like I wasn't able to contribute much as just one person. This may be compounded by the fact that I was just an intern, but it is still something to think about. I was offered a position as an intern for next summer, but I do not believe I will be accepting. I wish to gain experience in other areas of computer science more relevant to my interests, and will try my luck at another company next year.

References

[1] Bank of America — History. http://en.wikipedia.org/wiki/Bank_of_america#History. Accessed: 15/08/2012.

[2] Bank of America Merrill Lynch. http://en.wikipedia.org/wiki/Bank_of_America_Merrill_Lynch. Accessed: 15/08/2012.

[3] Former Merrill Lynch 450-acre campus in Hopewell to be sold under Bank of America plan.

[4] Merrill Lynch — History. http://en.wikipedia.org/wiki/Merrill_Lynch#History. Accessed: 15/08/2012.

[5] Robert Ballecer. This Week in Enterprise Tech: Ep. 3. Podcast, 2012.