TECHNICAL REPORT

Performance Testing of Mobile Applications with Multiple Technologies

Date submitted:

December 14, 2012

Principal Investigator:

Leonel E. Lagos, Ph.D., PMP®

Florida International University Collaborators:

Justin Phillips

Submitted to:

U.S. Department of Energy
Office of Environmental Management
Under Grant # DE-EM0000598

Applied Research Center
FLORIDA INTERNATIONAL UNIVERSITY

Abstract

Mobile devices are portable, battery powered devices running condensed operating systems normally referred to as mobile operating systems (OS). The term mobile device may also refer to tablets, touch input hybrid laptops, and other devices. At their inception, mobile devices maintained a relatively slow growth in consumer adoption and were considered a business or technology professional's tool. With the introduction of the touch interface by Apple in 2007, the mobile smart device made record breaking leaps in consumer technology adoption and usage. With nearly 50% of the United States' adult population owning smart phones, mobile devices are quickly overtaking traditional PCs as the most popular computing devices (Smith, 2012). Both businesses and government agencies are taking advantage of this change by enacting policies like "Bring Your Own Device" (BYOD); these policies lower the cost of both hardware and training as users are already familiar with their personal devices. Now that mobile devices are abundant in supply, many individuals come to expect support and access to government systems on the go and in the field. These devices are so packed with features, like GPS, cameras and accelerometers, that they provide a great tool for out of the office and other situations where a desktop PC is not convenient. Mobile devices are still a relatively new industry, with many platforms, hardware, and consumption methods to choose from. This lack of maturity within the industry and the number of incompatible platforms create a complex development task.

During the development of the mobile application subtask, it was realized that a significant amount of research in the technology, platform, development processes and framework, etc., is needed for future D&D KM-IT mobile development. Continued research work was needed to establish the best development process for mobile systems which can be implemented in all the future modules of D&D KM-IT. This research included many of the current industry standard mediums for data consumption over mobile devices. Three mobile application methods were chosen and developed in parallel. The methods tested were JavaScript asynchronous data requests via the jQuery library, web service request utilizing ASP.NET forms written in C#, and a native application running on Google Android written in Java. The development time, complexity, security, and performance were analyzed and recorded in order to determine the best technology for the task of producing the D&D KM-IT mobile module.

Table of Contents

Figures and Tables

1.0 Background

1.1 Introduction

A mobile device, sometimes referred to as personal data assistant, hand-held, or smart phone, is a small, hand-held computing device that a user typically interacts with through touch input and/or miniature keyboard. Apple, HTC, LG, Motorola, Nokia, RIM, and Samsung are a few examples of mobile device manufacturers, though there are only four major mobile operating systems (OS) that run on these manufacturers' hardware. These handheld devices may run an operating system that allows for the installation of 3rd party applications (apps). For connectivity, most devices can be equipped with WI-FI, Bluetooth, GPS, and cellular data radios, which allow for apps (web browsers, media players, etc.) to utilize network connections. These devices may also be equipped with a camera or media player feature for video and music. Mobile devices are usually powered by a lithium battery, which makes it necessary to utilize many power saving features; for instance, short screen timeouts, performance scaling, and throttling.

There are approximately 2.4 billion internet users worldwide; one billion of those are mobile users. There are now over one billion mobile broadband subscriptions worldwide, and mobile is set to be the access platform of choice for most people in the developing world, where fixed line penetration remains low (Parkes, 2012). Today, approximately two million smart phones are activated each day with the usage of mobile devices expected to overtake the personal computer (PC) over the next few years. With the launch of the iPhone in 2007, including a centralized application catalog, mobile devices were made more accessible to the general public. The introduction of a centralized application catalog allowed for general users to embrace applications besides those pre-loaded on the device. Most major mobile OSs now ship with a central application catalog. Additionally, most modern mobile OSs are touch-centric, utilizing icons for application navigation and minimal hardware buttons.

In 2010, with mass adoption of tablet computing devices and larger mobile device screens, the World Wide Web (web) began to see more rich mobile enabled websites. Prior to 2007, mobile browsing outside of the corporate environment was nearly non-existent. The introduction of touch input devices, larger screens, and the enabling of browser standards, as well as cookies, empowered mobiles to become a large website consumer group. Today, HTML5, along with hybrid applications, are making the distinction between native application and websites less obvious on both mobile devices and traditional PC.

2.0 Mobile Challenges

The convenience and accessibility of mobile devices make them a popular choice for non-technical users; however, the personal computer still has a considerable number of advantages over mobile devices at this time. The foremost of these advantages is the bandwidth limitation on mobile devices, both in throughput and allotment. Most carriers are in the midst of proliferating their new 4g cellular radio networks, about the bandwidth of DSL, though only when the device has full reception and when not competing for bandwidth with nearby users. Also, many cellular data providers have recently implemented bandwidth allotment caps on users on a monthly basis, though this allotment is not comparable to home broadband usage as mobile content is optimized for both screen size and throughput (Congressional Internet Caucus Advisory Committee, 2012). Security, not only the security of the software, is another area of concern with mobiles. There are many other security issues, discussed in Section 4.0, since these devices are mobile and many government and corporate entities have enacted “Bring Your Own Device” (BYOD) as a mobile policy.

The two other major hardware limitations aside from bandwidth are human interface and power consumption. Mobile devices are small in form, thus can be difficult to work on over long periods of time, having smaller screens and touch centric input. Recent trends, motivated by user preferences towards carrying less, have focused on integrating many different applications in a single general-purpose device, often resulting in much higher energy consumption and consequently much reduced battery life (Robert N. Mayo, 2003).

3.0 Mobile Platforms

Currently, there are four major mobile operating systems (OS). They are Android, Blackberry, iOS, and Windows 7. While the four major mobile operating systems are similar in user interface, they differ greatly in design and architecture. Fortunately, the various hardware and operating system differences do not carry over into the application development process. Each of the four operating systems do provide different development languages for their applications, but the overall development process and design is very similar across devices. Figure 1 provides a worldwide smartphone market forecast by OS.

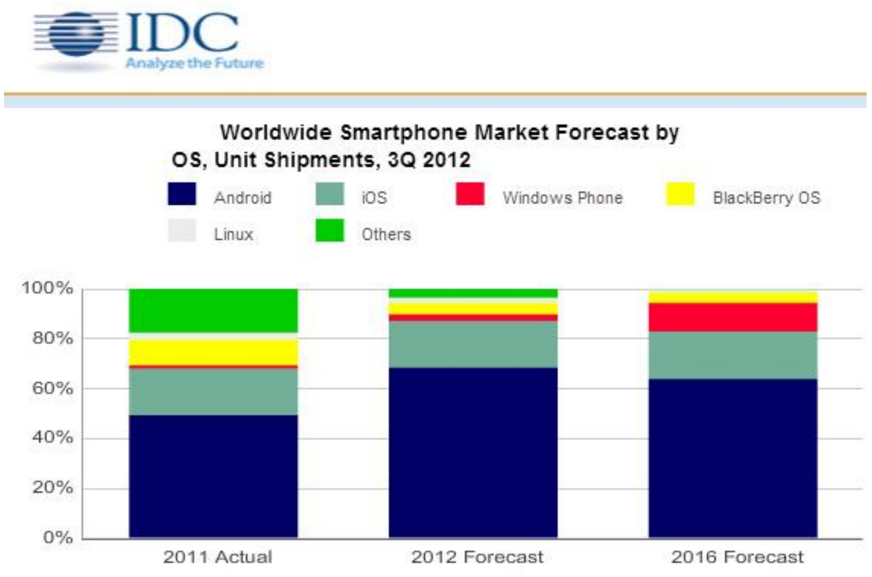

Figure 1: Worldwide Smartphone Market Forecast by OS, Unit Shipments, 3Q 2012

3.1 Native Client Applications

Native applications installed locally on the mobile device are coded with specific programming languages (Objective C for iOS, Java for Android). These mobile applications are fast, reliable, and powerful but are tied to the mobile platform they were developed for. That means one must duplicate them using the appropriate programming language in order to target another mobile platform, though there are many new programming practices to help reduce the immigration of code to other platforms. Native apps are usually more responsive, can access more hardware features on the device, and feature a richer user interface than web apps. However, the trade off is greater development cost and time as they require more testing and duplication of code.

3.2 iOS

iOS (previously iPhone OS) is a mobile operating system developed and distributed by Apple Inc. Originally released in 2007 for the iPhone and iPod Touch, it has been extended to support other Apple devices such as the iPad and Apple TV. Unlike Microsoft's Windows CE (Windows Phone) and Google's Android, Apple does not license iOS for installation on non-Apple hardware. As of June 12, 2012, Apple's App Store contained more than 650,000 iOS applications, which have collectively been downloaded more than 30 billion times. It had a 16% share of the smart phone operating system units sold in the last quarter of 2010, behind both Google's Android and Nokia's Symbian. In May 2010, in the United States, it accounted for 59% of mobile web data consumption (including use on both the iPod Touch and the iPad).

iOS manages the device hardware and provides the technologies required to implement native applications. The user interface of iOS is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons. The response to user input is immediate and provides a fluid interface. Interaction with the OS includes gestures such as swipe, tap, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS operating system and its multi-touch interface. Internal accelerometers are used by some applications to respond to shaking the device (one common result is the undo command) or rotating it in three dimensions (one common result is switching from portrait to landscape mode) (Apple, Inc., 2011). Apple’s iOS is sometimes referred to as a walled garden, as the only way to install applications is via the central application catalog named “App Store”. At this time, Apple does not support any other method, outside of their development tool set, to obtain 3rd party applications. Apple’s iOS is the second most popular mobile device OS.

3.3 Android

Building on the contributions of the open-source Linux community and more than 300 hardware, software, and carrier partners, Android has rapidly become the fastest-growing mobile OS. To date, there are more than 200 million activated Android devices. Android’s openness has made it popular for consumers and developers alike. In the third quarter 2011 alone, people downloaded more than 2.4 billion apps to their Android devices (Google, 2012).

Android, like iOS, includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language as well as the device’s hardware functionality. Similar to iOS, Android user input is centered on a multi-touch interface using gestures to interact with applications. Android features a central application catalog called “Play” (formerly “App Market”), though Android also supports the use of 3rd party application catalogs as well as app installations via browser or external storage. Android has been the most popular mobile device OS since 2010 when it overtook the iOS.

3.4 Windows

Windows Phone 7 operating system was released in 2010 and focuses on ease of use and simplicity. Windows 8, which was released in the fall of 2012, presents a unified tile system across mobile devices and traditional PCs. Windows mobile is one of the more simple operating systems; it trades customization and personalizing features for ease of use. Again, like iOS and Android, the Windows 7 operating system utilizes gestures for interaction with the application. Windows 7 ships with Microsoft's application catalog, which is called "Windows Marketplace".

3.5 BlackBerry

Formerly the OS of choice in terms of mobile devices, RIM's BlackBerry features an experience similar to the other OS mentioned. Blackberry utilizes a much more open 3rd party application catalog featuring applications developed across a large selection of platforms, including Android and HTML based apps. RIM popularized mobile devices in the corporate world by pioneering push technology within ergonomic form factors. All Blackberry devices, since 2009, have shipped with a central application catalog called "App World". RIM's Blackberry OS is the third most popular mobile device OS followed by Microsoft's Windows 6.5 and 7 for mobile devices.

4.0 Security

Security has been a concern on mobile devices for some time now for various reasons. They function with different, new operating systems. There are a range of security issues including: application update policies for both system and 3rd party applications is extremely limited, the four major operating systems all have many different versions with different security techniques, mobile devices often function on cellular data radios which are not controlled by internal IT departments, etc. One of the more unique concerns is the size and mobility of the device; they can be misplaced, lost, or even taken home, while programs like BYOD complicate the issue further by having a device function as both a personal and a work device. These issues can compromise the data stored by native applications and also web applications. Because of this, extra planning and development must go into mobile applications that would not normally be needed in a PC application (Accenture, 2012).

5.0 Application Development Process

Each of the platforms for mobile applications also has an integrated development environment, which provides tools to allow a developer to write, test, and deploy applications into the target platform environment. These tools usually include a software emulator for each platform to provide rapid initial testing. Third (3rd) party applications are sometimes pre-installed on mobile devices during manufacturing, can be downloaded by customers from various mobile software distribution platforms, or delivered as web applications over HTTP as web sites which use server-side or client-side processing (e.g. JavaScript) to provide an "application like" experience within a web browser or hybrid application.

The first step in mobile application development is to build the list of requirements. This should be an exhaustive list of requirements, beyond that of the client's request as mobile devices have more states, security issues, and resource limitations than most other computing devices. For instance, if a device goes into standby mode (powering off of the LCD and lowering of the processor performance), should all sensitive data be wiped and a new login be required. The developer does not know if, when the device comes out of standby mode, it will still be in the same office, with the same user. Following the development of requirements should come documentation. Documentation is always crucial in software development; however, it is even more so with mobile development because you may have multiple teams working on the same exact application, only for different platforms. Proper documentation and in code commenting will greatly ease cross platform integration and keep teams from duplicating their efforts.

The next stage of the process is designing the mobile interface and flow. Mobile interface design is key to creating a high quality application. The designer must take into consideration the touch interface, high volume of non-technical users, small screen size, and device specific layout standards which make for a very different design experience. Users expect an easy to navigate application that is visually similar to the OS they are using. The design should be documented and tested with a wide selection of users from different devices to insure functionality and ease use.

The development stage of the mobile application is similar to most software development. Each platform developer provides the tools necessary to create the user interface and application code, as well as debugging tools and emulation. Utilizing a very high standard of object oriented coding is essential when building applications for multiple platforms. The application is best split into three areas, user interface, business logic, and platform framework, because the user interface and platform framework are usually very different from platform to platform. There are tools that allow developers to create applications for most of the mobile device platforms from one code base, though these tools are not yet mature and can cause many hardware specific issues. Web applications outperform native applications in development because they are based off of one standard and display the same on each device.

Once the initial development phase is complete, the application is usually alpha tested, this usually consists of the developers, designers, and other project members testing the application for code related bugs and crashes. Following the alpha test, a beta test is conducted which encompasses a broader range of users and looks for bugs within the business logic and over a wider range of hardware. Most development firms make use of a bug tracking system, like Microsoft Team Foundation Server, to track bugs, collect users' stories, and assign fixes.

Once the testing is concluded and all documented issues are resolved, the application is ready for approval and release. The release scheduling is often frustrating for clients and developers because each platform has a separate review process for approving the application; thus, it is difficult to estimate the exact release date. Once approved by each platform application catalog, the application is hosted on the catalogs' servers and bound by the rules of that catalog. Once live, the catalog's users may install, comment on, and rate the application.

6.0 Mobile Web Application

Mobile web applications are increasingly popular due to their lower development time, cost effectiveness, and more definitive control over consumption and releases. Industry best practices recommend mobile web applications and hybrid native applications unless an application requires access to certain device hardware or additional graphics. Modern mobile devices ship with browsers that adhere to common web standards; thus this form of development does not require multiple projects for one application. This also allows for complete control over the application by no longer requiring adherence to application catalog rules and allows for direct access to the application that users are interacting with. While the performance of graphical heavy interfaces is not the same as native applications, HTML5 is helping web apps perform better and even be usable while offline. The mobile web application can even be exposed via native applications using a hybrid technique. The hybrid development takes the best of both worlds, the ability to advertise and consume the application via the central application catalog and the ease of having one codebase with only minor source code for each mobile operating system, as well as more control over releases and consumption (Cavazza, 2011). Though the web application approach provides a unified code base, there are many approaches to the design and development of such an application.

In addition, a great strength of web application is the ability for them to be exposed to the internet and reachable by search engines. This capability makes the web application more popular across mobile platforms because it does not limit the usage to a single platform. The content within the web application can be consumed by anyone seeking information via search engines. This feature is unique to web applications and it opens other marketing and sharing opportunities available to the web user like the ability to share the link via social media networks, including email.

7.0 Methodology

Three applications were developed in parallel for recording and comparing the performance of each major method or solution to mobile devices interfacing with the D&D KM-IT (Figure 2).

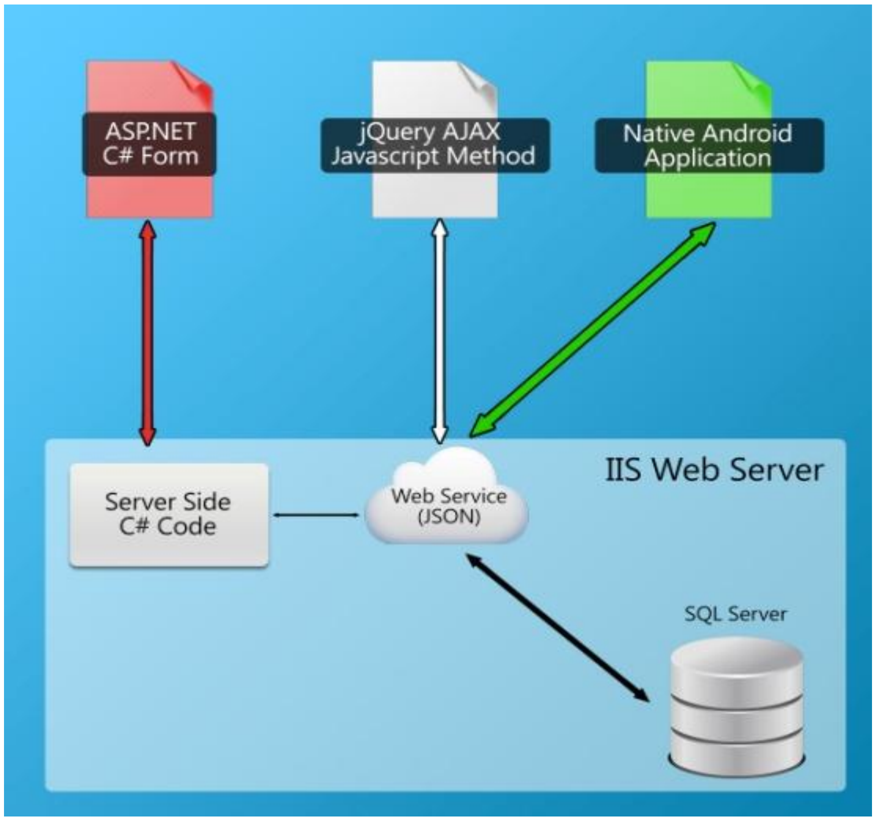

Figure 2: Client- Server interaction between different development methods

Each application records the time it takes to execute certain stages in the querying for data and then submits the results to a database, where analysis and comparison can be performed on these results. Each query utilized the technology module, requesting the title, description, and id of every technology listed in the Technology Module; approximately 523 technologies. This data was then returned in JSON format by the web service and was approximately 140 kilobytes in size. Because each application queries the data in a distinct way, it is impossible to compare the data directly, though it is still important to record the many stages of a request's life cycle as this data is helpful in determining the cause of performance variance in the testing. Over 12,000 tests were performed and recorded. These recorded stages are as follows:

- **Page Load**
  The page load in ASP.NET C# Forms captures the time before controls and elements are built. This time was set to zero milliseconds and all of the following stages are based on this time.
- **Send Request**
  This stage is the amount of time between the page load event and the initialization of the request for data from the web service.

- **Receive Response**

  The time captured at this stage is the exact moment that a response is received from the web service containing the data queried.
- **Process Response**

  The time captured here is that of the moment the serialized data received in the response is de-serialized and placed into an object or collection of objects.
- **Render Response**

  The render stage happens once the data begins being rendered for the user to view.
- **Final Page Result**

  The final page result happens once all data has been queried, received, and rendered. This is the end of the page's life cycle.

7.1 jQuery AJAX Application

The first of these applications developed utilized JavaScript's AJAX protocol to query the web service and display results. In this method, the client browser makes calls asynchronously while the user is browsing a page. This is a very popular method used by Facebook and many Google applications. It enables faster response times by consuming less bandwidth due to only data being requested, versus each page request calling the user interface elements. The major differences with this method is that queries to the web service are made while the page is rendering; the majority of the applications execution happens on the client hardware, and further requests can be initiated without reloading the page or user interface elements.

On desktops, this method performs extremely well due to the processing abilities of modern desktop process coupled with a set of highly optimized JavaScript engines utilized by desktop browsers. However, on mobile devices, the processing capabilities are not the same and in turn create less performance. The results are also variable depending on the mobile browser used as many mobile browsers use JavaScript engines that are poorly optimized for the hardware they are running on.

7.2 ASP.NET C# Web Forms Application

ASP.NET Web Forms utilize server side processes to execute the majority of code, supplementing this with minimal JavaScript on the client browser. This method executes the request, receives the response, processes the data, and prepares the rendered code all on the server, which requires more performance out of the server, but far less from the client. C# is a high level language that has many classes and features built-in, requiring less coding than JavaScript and a stronger typed object allowing for smoother debugging and development. In theory, this method should perform well on mobile devices since it requires so little code execution on the client browser. However, requests are heavier as each request contains JavaScript code, user interface elements, and page state information as well as the data queried.

7.3 Native Android Java Application

Java is an equally high level language as Visual C# for supplying the ease of debugging and coding environment. A native Android application executes all code locally on the Android OS, utilizing disk space and local libraries. The Java application spawns a new thread in the OS for each query of the web service, providing a highly optimized performance. The Java execution environment is highly optimized for the hardware it runs on and so theoretically should provide the best performance as these are multi-threaded asynchronous applications. Being that these applications reside and execute locally in the OS, they require more security and state management; the application must know what to do if the phone rings, a text is received, the user puts the device into standby, etc. Also, distribution of the application is much more difficult because the prime method for distribution is a proprietary, hands off, application catalog. This, coupled with the fact that Android is only one of four major mobile OSs, greatly increases the development time, cost, and testing. Android was chosen as the 3rd test application to represent native device applications. Android is currently the most popular OS for mobile devices and native applications when it comes to non-media based UI performance. (Bhatia, 2012)

8.0 Mobile Hardware

To fairly and accurately gain perspective on mobile performance, a wide range of hardware and OS were selected. Used in these tests were: Apple iPod Touch, Apple iPhone 5, Apple iPad 2, BlackBerry Bold, HTC Radar Windows 7.5, HTC HD2, LG Optimus S, Samsung Galaxy Nexus, and Samsung Galaxy Tab 2. This set of devices takes into consideration the wide range of hardware and OS fragmentation of Android and the wide range of hardware on iOS as well as other popular OS. Performance varied significantly between devices.. Their specifications did not match their performance in some case. For instance, the iPhone 5 has one of the fastest processing units and graphical capabilities, but the

Samsung Galaxy Nexus (a relatively older device with a much slower processor) out performs the iPhone 5 in some browser rendering, thus giving the Galaxy a better result.

To ensure that the tests results accurately displayed each OS and method's actual performance and not that of multiple types of wireless radio technology, all tests were conducted via WIFI using the 802.11 G standard, each device maintained excellent reception throughout the tests. Also, the default browser of each operating system was used for each device according to the majority market share that default browsers hold. This excludes Android which has a more distributed browser market share and so other browsers were also tested on the Android OS.

9.0 Results

Just over 12,000 tests were run, evenly distributed across the four mobile OS and methodologies used. The two most obvious results were that the native application is two to three times faster in response time than the browser based applications and that both the native application and ASP.NET application had consistent performance. The jQuery AJAX method was the least consistent method, especially across OS and on the lower end hardware. From a purely performance perspective, it is clear that the native Android application is superior in speed, consistency, and dependability, with ASP.NET not far behind in terms of dependability and consistency.

A mobile device's rendering performance on the browser is heavily dependent on the hardware, where as with modern desktops it is very difficult to notice the difference in page rendering performance between high and low-end machines. Current generation iOS devices performed very similarly, however Android devices, which ranged from low to high-end had a broader spread of results. This disparity is a product of the price point difference between the two operating system's hardware; there are no low-end iOS devices. Therefore by including a range of Android devices, the Android averages as a whole do not accurately represent any one Android device. Instead, they show the median results of a collection of devices.

JavaScript's dependence on the mobile device's processing power hinders its performance and consistency greatly. Though the native Android application calls and processes the data in much the same way (locally on the device), the Java execution is much more optimized and runs at a lower level than JavaScript, so the performance varies greatly between the two methods. Mobile devices are complex tools that have a great many tasks going at once; for instance, at one point one of the Android devices was updating applications in the background while the tests were running. This lowered the jQuery AJAX performance by nearly 300% on that device. While this method is a very efficient and effective way to query data from a traditional PC, the energy saving focused mobile devices are not as up to the task.

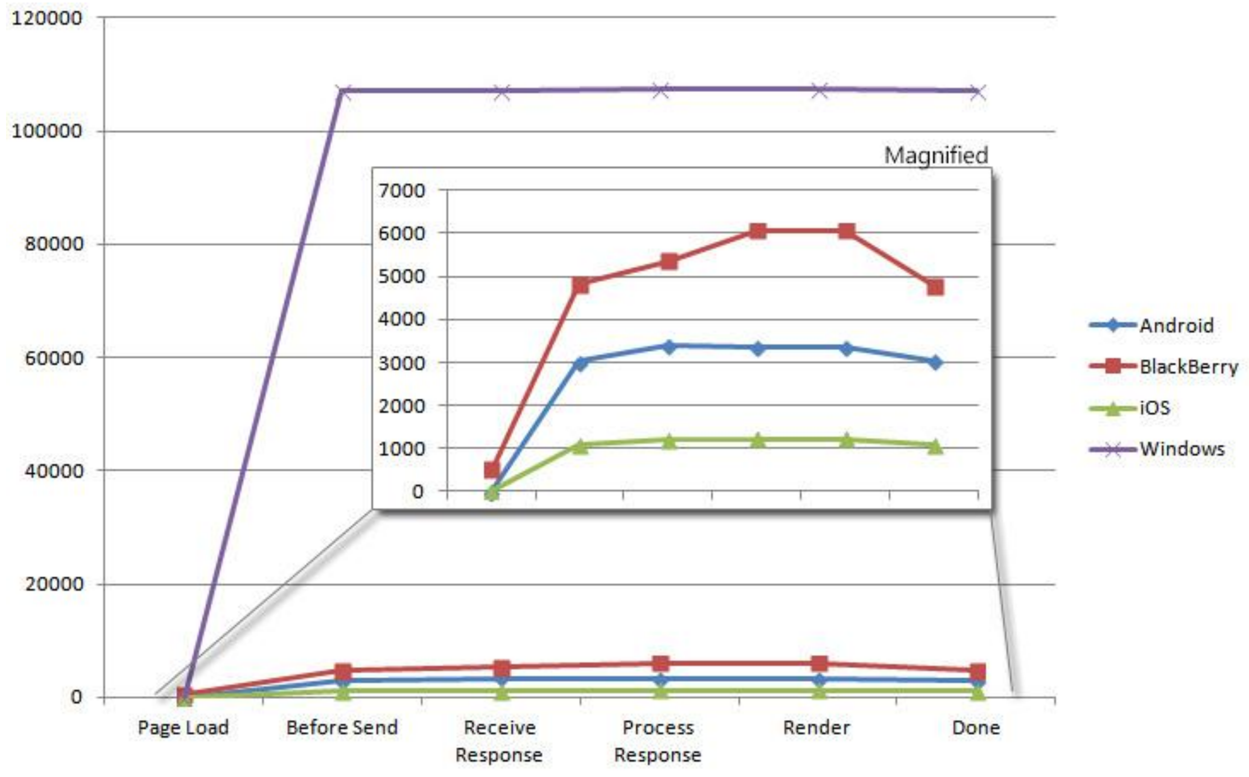

Figure 3: jQuery Ajax performance comparison, units in milliseconds

The ASP.NET C# application's ability to query and process data on the web server gives it an advantage over the other methods because it only needs to render the HTML provided by the server. This method may have inconsistencies when the server is under a greater load, or the server is running other background processes, but in comparison, the server has much greater processing power and runs much more optimized lower level code that can be scaled and cached. To help simulate heavier loads, many of the 12,000 tests were done within less than one second of each other. At one point, the server received a request every 0.5 seconds over the duration of one hour, resulting in half of the tests conducted.

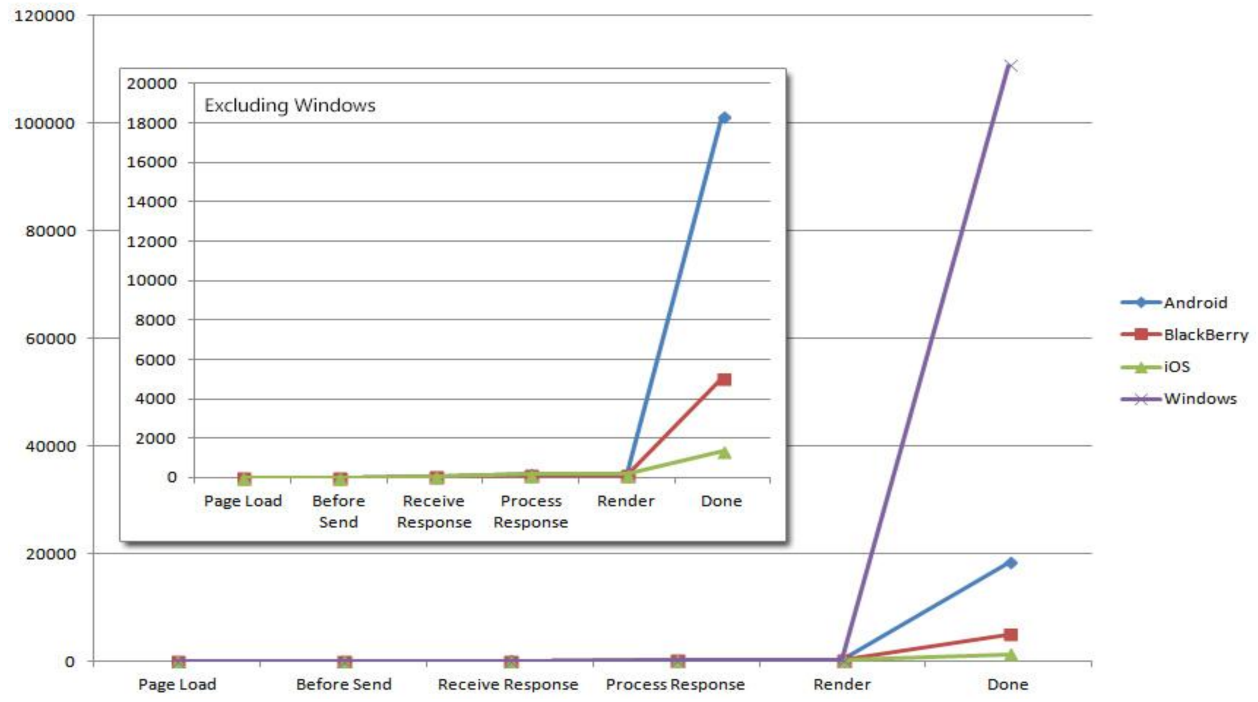

Figure 4: ASP.NET C# performance comparison, units in milliseconds

Note that not all of the 12,000 results consisted of bad data. Some inconsistencies and device specific issues made certain records unusable.

9.1 Inconsistencies

During the testing process, there were several inconsistencies worth mentioning. These resulted in variations of data. However, on each and every case these irregularities appear, there was also an event associated with it that supported the result. Each of the platforms had one form of limitation, event or process that gave way to some of the out bound data points in the result. For instance, on the mobile windows devices, the JavaScript engine performed slowly at times and the page load event for JavaScript begins execution far later than on other devices. The JavaScript engine is critical for web applications because the majority of web applications count on JavaScript to perform the heavy lifting on the client device. If the JavaScript runs slowly or locks up, then it will have a big impact on the results. Our testing experienced this scenario several times. Other inconsistencies on the data occurred because of AJAX's cache issues. As stated in section 7.1, AJAX makes a call to the client in the background and can target specific element on the page. The advantage is that it caches the data so the user does not experience a complete reload of the page they are viewing. This technique is unique to web applications and it can improve the response time of any web application. However, since AJAX is dependent on

JavaScript, if there is a problem with the performance of the JavaScript on the device then AJAX will be directly affected as well.

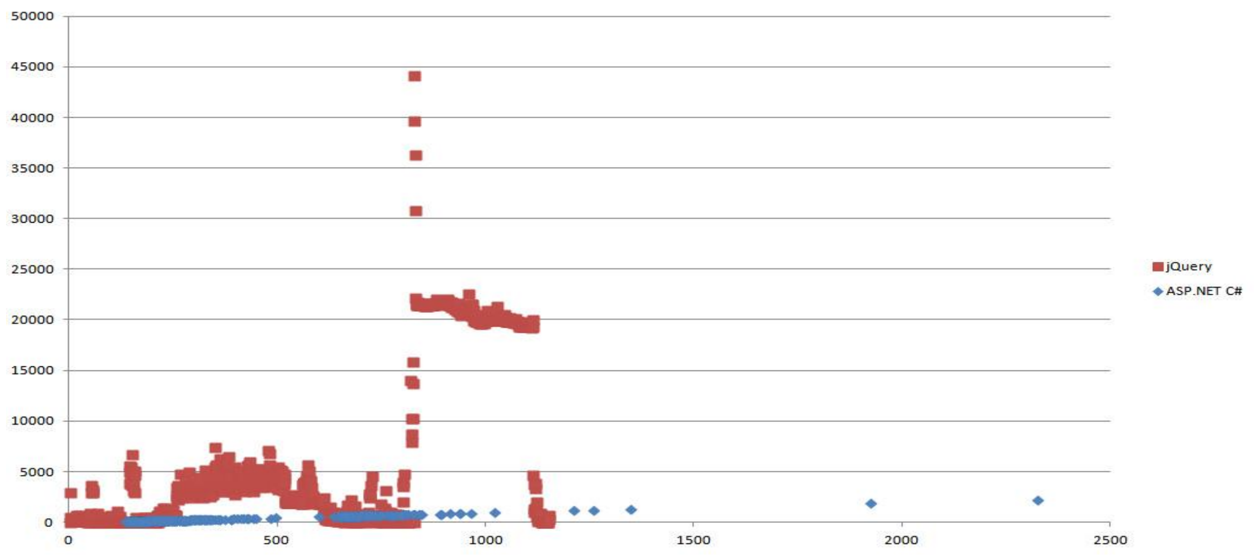

Figure 5: Comparison of Android JavaScript hardware inconsistencies, units in milliseconds

Finally, other issues related to the devices' operating system involved device timeouts and updates. Each operating system has setting used to optimize the device. One of the settings is device timeout. They include a time out for screen display, browsing session, call management, application response, etc. Many of these timeouts settings are not accessible to the user and are built-into the operating system. These timeout sessions made variations in our results because there were instances where the device needed to establish a connection to the server and this increased the time it took to gather the first sets of data. We also experienced, in one or two cases, a device that performed an automatic update in the middle of a testing session. These updates are randomly done by the operating system and 3rd party application installed on the device. CPU and memory resources are shifted and reallocated within the device to accommodate this multitask and those affected any testing session performed at the time.

10.0 Conclusions

There are many approaches to building mobile applications; an important aspect of deciding which approach is best for a particular project is identifying the requirements for graphics, data storage (offline usage), security, target devices, and available resources. Our evaluation duplicated these methods in order to determine the best platform for future modules of D&D KM-IT Mobile. Three mobile application methods were chosen and developed in parallel. The methods tested were native application running on Google Android written in Java, JavaScript asynchronous data requests via the jQuery library using AJAX and web service request utilizing ASP.NET web forms written in C#.

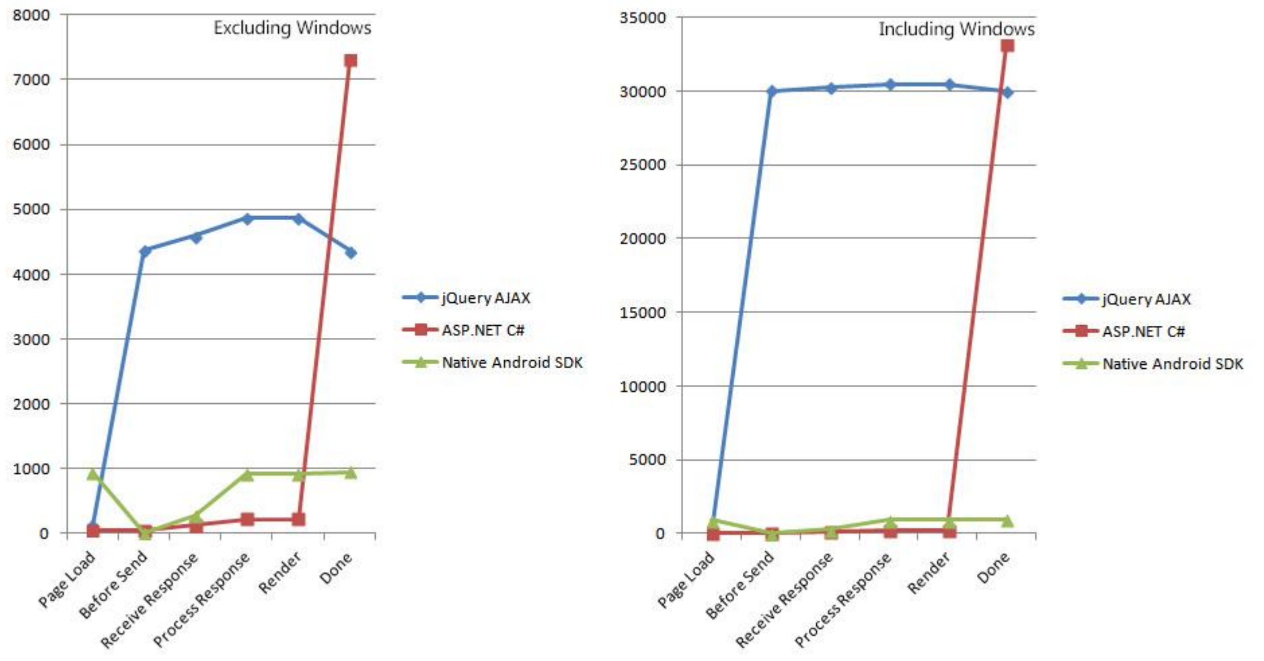

Figure 6: Comparison of all methods used with units in milliseconds

The native application design showed promising results in performance. However, we concluded that there were significant issues to consider. To start, the application would have to be available using the respective App Market (e.g. App Market, Google Play Store) for each of the operating systems. This is convenient for each user to install but it limits the application to that operating system unless multiple versions of the application are developed for each App Market. Right away the development cost will increase because multiple applications means multiple skill sets of the developers and synchronization between the platforms to make sure their capabilities all match. Being part of an App Market also means that the developers have to abide by their rules which may be strict at times. The application also have to be kept up to date at all times to stay current with the operating system where is running. Finally, the application won’t be available to search engines from a browser, only within the App Market, limiting the exposure to potential users.

The results of the JavaScript JQuery application using AJAX were inconsistent. The application responded well with high end mobile devices like the Samsung Galaxy Nexus, Samsung Galaxy Tab 2 and iPhone 5.

Other devices had serious performance issues like the LG Optimus S, and even the windows devices experienced problems because of the slow JavaScript rendering related to the operating system. On the windows mobile device, this method performed very poorly even though the device was high-end simply because of the way the operating system handles and gives priority to JavaScript. In general, it was noted that the application performance varied significantly since JavaScript runs on the client device and is limited to the resources of the device. The development time for this type of application is longer that ASP.NET, for instance, because the language is always changing and AJAX does not have a solid class library that developers can use. In this scenario, developers would have to write extra code when compared to a typical ASP.NET application.

The ASP.NET C# Web Form application performed respectably well across all devices. The web application ranked high in performance, stability and development time. The windows device executed similarly to the performance it had with jQuery AJAX, this is due to the Internet Explorer 8 browser and it is considerably slower to load and JavaScript performance compared to the modern browser on the other devices. This result was almost expected, but in fact, all the devices performed better under this method. The development time and maintenance of the application is a huge factor as well. Under ASP.NET Visual C#, developers developed one application that can be viewed by multiple devices. The only requirement was that the device would have a browser which came standard as part of each device's operating system. The ability to have one "project" for multiple devices makes the application more efficient and lower the development cost greatly. The ASP.NET web application is not dependent of an App Market for deployment. It can simply reside on the internet and be available to anyone with a browser which also makes the application reachable by search engines. The application can be shared with social network and email which also greatly increases exposure. Web applications can be secured by the hosting company policies and other tools like Secure Socket Layers (SSL).

11.0 Recommendations

The findings within this research conclude that based on the D&D KM-IT Mobile requirements, future modules will benefit the most from using the ASP.NET Visual C# Web Forms Application.

There are many approaches to building mobile applications; an important aspect of deciding which approach is best for a particular project is identifying the requirements for graphics, data storage (offline usage), security, target devices, and available resources. Since the Applied Research Center's mobile knowledge management application does not require intense graphics or offline availability, a reliable web application would be the first choice for most development teams, as the native application requires four separate codebases, longer development times, and much more maintenance. The ASP.NET Visual C# approach gives the developers more control over releases, streamlines development, and provides a similar environment to the desktop tools (D&D KM-IT) while also providing good performance with consistent and dependable functionality. An added benefit to providing a mobile web application is that they are exposed to search engines and consumable by third party applications. In contrast, native applications require installation via an application catalog before consumption and their contents cannot be accessed via search engine.

Many mobile applications fall into the hybrid or web application approach. It is essential to make an exhaustive review of the applications requirements before choosing any approach as each is radically different in terms of development practice, cost, and time. It would be a severe setback if a team were to begin developing a web application for offline usage, only to realize some of the mobile devices they are targeting do not support HTML5's offline capabilities. Lastly, it is important to note that the mobile technology industry is extremely volatile; the standards for native applications, their capabilities, and their frameworks are constantly in flux as the market evolves.

If the team decides to develop a web based application, then the next stage is to choose the type of web development platform. Two popular implementations are ASP.NET Web Forms and ASP.NET MVC. ASP.NET Web Forms, being the more popular method, has less initial investment and more rapid development practices. ASP.NET coupled with jQuery Mobile allows developers to rapidly develop and design rich application interfaces that work across platforms. AJAX can be added to these web applications to help conserve bandwidth and provide the user with a smooth experience; however, this requires considerable forethought and development time because the majority of the user interface is developed in JavaScript. Also, many AJAX techniques are not supported by all mobile devices so it is best to implement them with caution.

12.0 References

Accenture. (2012, 02 03). *Mobile Application Development: Challenges and Best Practices .* Retrieved from Accenture: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Mobile-Application-Development-Challenges-Best-Practices.pdf#zoom=50

Anderson, R. (2011, Sept 12). *ASP.NET MVC 4 Mobile Features.* Retrieved from ASP.NET: 2011

Apple, Inc. (2011, 06 05). *About iOS Development.* Retrieved 06 15, 2012, from iOS Developer Library: http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/IPhoneOSOverview/IPhoneOSOverview.html#//apple_ref/doc/uid/TP40007898-CH4-SW1

Bhatia, K. D. (2012, 12 03). *Thesis: Comparing Mobile Platforms.* Retrieved from San Diego State University: http://sdsu-dspace.calstate.edu/bitstream/handle/10211.10/1324/Bhatia_Kunal.pdf?sequence=1

Cavazza, F. (2011, 09 27). Mobile Web App vs. Native App? It's Complicated. *Forbes*.

Congressional Internet Caucus Advisory Committee. (2012, May 3rd). *The 2012 State of the Mobile Net Conference.* Retrieved June 20th, 2012, from Congressional Internet Caucus Advisory Committee: http://www.netcaucus.org/conference/2012/sotmn/

Google. (2012, 01 27). *Android Developers*. Retrieved 06 15, 2012, from Android: http://www.android.com/developers/

Parkes, S. (2012, June 15). *Broadband Commission Open Letter to G20: Bring broadband to the world.* Retrieved June 20, 2012, from International Telecommunication Union: http://www.itu.int/net/pressoffice/press_releases/2012/39.aspx

Robert N. Mayo, P. R. (2003). *Energy Consumption in Mobile Devices.* Palo Alto: HP Laboratories. Retrieved from Energy Consumption in Mobile Devices:

Smith, A. (2012, March 1). *Nearly half of American adults are smartphone owners.* Retrieved from Pew Research Center: http://pewinternet.org/Reports/2012/Smartphone-Update-2012.aspx