IT@Intel White Paper

Intel IT
Mobile App Development
January 2014

Learnings from Early Native Apps Improve HTML5 and Hybrid Apps

We identified several best known methods for native development, which we are now applying as we shift our focus from native functionality and features to newer emerging and maturing technologies.

David Byrne
Handheld Technology Specialist, Intel IT

Joseph Doolittle
Enterprise Architect, Intel IT

Patrick Salo
Apple Branded Client Engineer, Intel IT

Executive Overview

Over the past two years, Intel IT has validated and improved its mobile application development framework, which was built to accelerate the delivery of secure mobile enterprise applications to Intel employees. Focusing first on native development, we identified best-known methods (BKMs) that we used to refine our framework and applied these methods to emerging development technologies.

In 2011, HTML5 and hybrid mobile application development technology was immature. So we focused instead on native development, which offered a rich user experience and streamlined connectivity. At first, we attempted to port entire enterprise applications from laptops to mobile devices and encountered several challenges. These challenges included slow performance and lack of full VPN and LAN access on mobile devices.

We then conducted a pilot project to deliver increased productivity and flexibility to business travelers by enabling native mobile access to a specific functionality: our Air Shuttle application. In the process, we successfully overcame the technical challenges we had encountered during earlier mobile application development efforts. We also identified several BKMs for native development, which we are now applying as we shift our focus from native functionality and features to newer emerging and maturing technologies. These methods include the following:

- **Componentization.** Our practice is to componentize mobile applications and connect features to back-end services for all delivery mechanisms.
- **Reusable libraries.** We developed reusable native libraries and ported them to the Web/HTML5 and hybrid delivery mechanisms.
- **Creative solutions.** Shifting our thinking from desktop and notebook applications, we explore creative ways to meet mobile development needs.
- **Best delivery mechanism.** We consider the business needs of the mobile application, weighing the advantages and disadvantages that each delivery mechanism provides.
- **Third-party integration.** Taking our lessons learned in connectivity and security, we educate our third-party suppliers on how to integrate with our back-end services while traversing our firewall.
- **Collaboration.** Working together to streamline mobile application development, we collaborate and benefit from each other's work.

By having one strategic direction and common resources, we are able to eliminate duplicate efforts and improve the process overall.

Contents

IT@INTEL

The IT@Intel program connects IT professionals around the world with their peers inside our organization – sharing lessons learned, methods and strategies. Our goal is simple: Share Intel IT best practices that create business value and make IT a competitive advantage. Visit us today at www.intel.com/IT or contact your local Intel representative if you'd like to learn more.

BACKGROUND

The consumerization of IT and growth of bring-your-own (BYO) devices have changed the way Intel employees work and what they expect from IT. Employees now want to be able to access enterprise applications and services from any device and any location. Intel IT saw the need to develop a cross-platform framework that would enable us to develop and deploy mobile applications on various form factors and OSs.

In early 2010, Intel IT implemented a bring-your-own-device program that allowed employees to use their own smartphones to access corporate data. On these devices, employees have evolved from using the provided calendar, contacts, and email services to expecting more access to enterprise services and applications while on the move. In response, we have provided increased access to information and IT services to improve flexibility and boost productivity. Internal data indicates that employee productivity and job satisfaction have increased as a direct result of implementing personal devices in the enterprise, with a user satisfaction rating of 94 percent (measured in the fourth quarter of 2012). Employees send approximately 2.27 million business-related email messages each quarter from corporate and personal devices. More importantly, employees report a time savings of about an hour per day by using personal devices.[1]

As shown in Figure 1, mobile devices have grown to over 42,200 corporate and personal devices in our current mixed OS environment.

Intel's Mobile Application Framework

Responding to the increased user demand for mobile enterprise applications, Intel IT formed a cross-organization team that began developing a cross-platform mobile application framework in early 2011. This extended framework enables us to provide a fast-track mobile application development and deployment capability that can keep pace with the rapid introduction of form factors, interaction methods, and OSs. This solution also provides mitigating capabilities and the ability to grant or revoke appropriate access levels, which address the common challenges that our enterprise IT department faces with respect to data security, privacy, and device manageability.[2]

[1] See "Best Practices for Enabling Employee-owned Smart Phones in the Enterprise," December 2011.

[2] See "Implementing a Cross-Platform Enterprise Mobile Application Framework," July 2013.

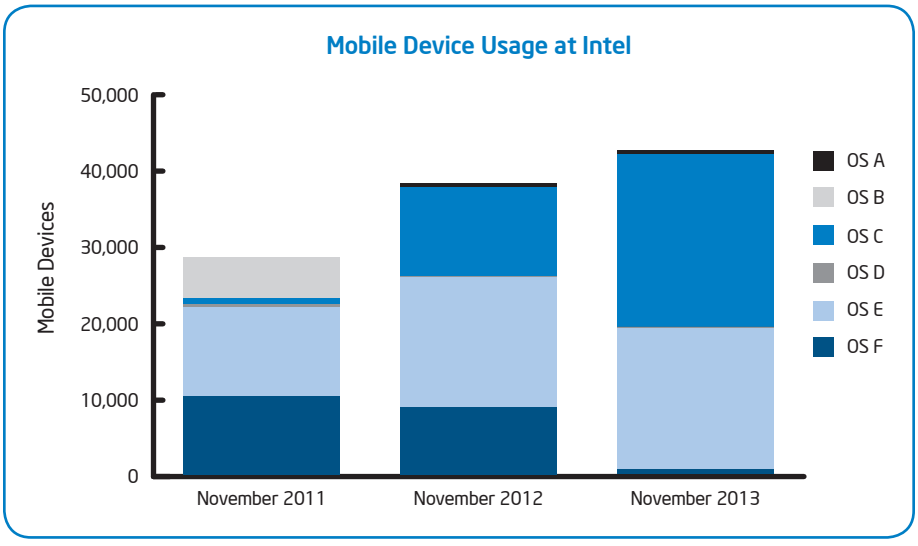

Figure 1. From 2011 to 2013, mobile device usage at Intel has nearly doubled in our mixed OS environment.

Development of a cross-platform mobile development application framework is part of our larger effort to enable enterprise applications to support the devices of today, such as touch-enabled, Intel® architecture-based business Ultrabook™ devices, 2-in-1 devices, and tablets, and to develop applications that support emerging interaction methods, such as voice, gesture, and perceptual computing. (See the sidebar, "Transforming the IT Ecosystem to Support the Applications and Devices of the Future".)

The mobile application framework supports several ways for deploying mobile applications, as described in Table 1.[3]

Early Focus on Native Delivery Mechanism

As we began our early efforts to mobilize enterprise applications, we evaluated the native, Web/HTML5, and hybrid delivery mechanisms. Each mechanism had pros and cons associated with user experience (UX), ease of connectivity, time to deliver, and level of reuse across mobile platforms. In 2011, HTML5 was still an emerging technology that often provided a less-than-optimal UX and application functionality. Also, we had not yet selected a mobile enterprise application platform (MEAP) provider. Therefore, during that phase, we focused on the native delivery mechanism based on its relative maturity and its superior UX.

Native development enabled us to focus on solutions that met corporate legal, human resources, and security policies; however, it limited our ability to reuse the code base on multiple OSs. We concentrated on solving the enterprise requirements and developing best-known methods (BKMs) so that later we could apply these learnings to the other delivery mechanisms.

Table 1. Advantages and Disadvantages of Mobile Application Deployment Mechanisms

Deployment Mechanism	Advantages	Disadvantages
Native Applications are platform-dependent Applications are delivered by an app store and updated per device, per OS	▪ Often provides best user experience ▪ Provides access to device sensors and features ▪ Enables rapid adoption of new user experience paradigms	▪ Requires recoding resources one OS at a time ▪ Requires technical support for multiple OSs ▪ Requires unique code for different screen sizes
Web Browser Only (HTML5) Applications are platform-independent Applications are delivered through the browser Most WebKit browsers are supported	▪ Enables code portability ▪ Provides optional coding and delivery mechanisms ▪ Increasingly supports device capabilities	▪ Relies on immature HTML5 implementation ▪ Often requires network connectivity ▪ Can require good network connectivity for optimal user experience
Hybrid (Native + Web) Applications are platform-independent Each container is secure with platform-agnostic content Each container is native to a device and uses HTML5 code	▪ Enables code portability ▪ Provides optional coding and delivery mechanisms ▪ Provides access to device sensors and features ▪ Increasingly provides access to device capabilities	▪ Relies on immature HTML5 implementation ▪ Often requires network connectivity ▪ Can require good network connectivity for optimal user experience

Transforming the IT Ecosystem to Support the Applications and Devices of the Future

We identified five criteria that lead to a better end-user experience—security, ease of use, platform independence, device independence, and support for emerging devices and interactions. These same five criteria lead to a better application developer experience by removing obstacles and providing readily available tools to increase application developer productivity and efficiency.

The implementation of a mobile application development framework enables us to create mobile applications that meet all five criteria. And, although the end result is applications that work better now and can take advantage of future technology changes, our work affects more than application development—it affects many components of the Intel computing environment, including security and privacy policies, mobile device management, mobile application lifecycle management, and application testing and scalability.

[3] See "Building a Mobile Application Development Framework," August 2012.

Early Mobile Development Challenges

Our early mobile development efforts focused on porting utility applications from notebooks to mobile devices. We started with the Intel Employee Portal, our website that provides an extensive set of resources for employees. This portal was initially available to employees using notebooks. Our goal was to make it accessible to smartphone users as well.

From this project, we discovered the challenges involved in attempting to mobilize an entire application. One significant challenge was that the performance on the mobile device was much too slow. While notebooks could handle large data sets, mobile devices lacked the processing power, local cache and storage, and screen size to handle the volume of information being returned. Compounding the problem was the connectivity at the time, which was available only at 3G network speeds. We were also constrained by the hardware and software available on mobile devices.

Another significant challenge was enabling VPN access on mobile devices. Notebooks connect over VPN to enterprise services with less restrictive infrastructure controls due to the mature level of device capabilities and overall device management. On mobile devices, open VPN tunnels were not allowed under Intel's security guidelines. We had to devise an approach for explicitly enabling only the desired services.

At the same time, other business groups were working in isolation and encountering these same challenges. We realized there was a need for sharing a strategic direction and common resources, so we could eliminate duplicate efforts and improve the process overall.

Table 2 describes the other technical challenges that we encountered.

AIR SHUTTLE MOBILE APPLICATION: A BETTER APPROACH

In the third quarter of 2011, we designed a pilot project to refine our mobile application development framework and to solve the technical challenges we had encountered so far. We decided to mobilize the Air Shuttle web application, which is used to make reservations on Intel's chartered plane service for travel between major Intel U.S. campuses.

IT and Corporate Services (the owners of the Air Shuttle web application) partnered to enable access to the Air Shuttle application on two types of native mobile devices. We also recently introduced an HTML5 version in September 2013. The scope of the Air Shuttle project is shown in Figure 2.

Table 2. Challenges Encountered When Porting Existing Applications to Mobile Devices

Technical Development Challenge	Description
LAN Access	On mobile devices, developers no longer had full connectivity and access to the LAN.
Environment	In most cases, developers had to switch PC platforms, change development environments, and tackle new challenges such as memory management.
Ecosystem	Developers were challenged by changes to the ecosystem itself, including different cultural norms, methods for development, and levels of toolset maturity.
Framework	The new mobile frameworks changed the existing daily development tasks.
Code Repositories	Mobile development used a different set of repositories than those that were typically used internally.
Quality Assurance (QA)	QA for mobile devices did not exist. The QA team did not have the tools required to simulate the mobile devices in our environment that were connecting through our infrastructure. Additionally, there were no client tools to validate good application practices (for example, testing the impact of the code on the device's battery life). These challenges restricted QA to performing just functional testing on limited hardware instances.
Corporate Asset Protection	Mobile applications required new ways of protecting data, which included application firewalls, VPN with whitelisted applications, and management of corporate credentials.
Legal	Because this was the first time we were deploying software to devices that were not owned by Intel or fully managed by IT, we needed to ensure that all legal issues were reviewed and approved. We also needed to ensure that user privacy was maintained.
Potential for Hacking, Breaches, and Abuse	Because some of the devices were not owned by Intel, we had no control over the OS revision, software installed, networks the devices were used on, and so on. Therefore, it was important to ensure that Intel intellectual property would remain safe and secure even though IT had limited control of the devices themselves.
Branding Guidelines	Internal branding guidelines needed to be developed and deployed.
Key Building Blocks	Several software methods needed to be developed internally for cross-product and cross-platform development.

From our past experiences, we learned that attempting to make an entire enterprise application available on a mobile device led to significant challenges, such as slow performance. This time, we concentrated on mobilizing specific feature sets. Frequent business travelers were asked to list the top priority items to mobilize, which were identified as the ability to book, view, and update flights and ground transportation, and to use the wait-list and flight status functions. Focusing on these use cases enabled us to break the development effort into smaller pieces.

The solution required web services to provide connectivity to the main Air Shuttle application database and to handle complex logic. A native mobile application was developed to provide a rich UX and to leverage the most mature technologies available for enterprise mobile application development at the time.

In November 2011, we successfully deployed the Air Shuttle mobile application by using the processes, governance, tools, and technology outlined in the mobile application development framework. Figure 3 illustrates how the application appears on a mobile device.

Application Results

From a business standpoint, the Air Shuttle mobile application project included the following highlights:

- 95-percent first-time successful provisioning to the device
- Over 5,000 downloads
- Used by an average of 500 users per week
- Increased productivity and flexibility for users

From an IT standpoint, the Air Shuttle mobile application achieved many technical firsts, including the following:

- The first internally developed mobile native application accessing services on Intel's intranet
- The first mobile application delivered utilizing a mobile VPN product
- The first mobile project to execute independent QA testing

Perhaps most importantly, Air Shuttle was the first application to follow the mobile application development framework and iteratively improve the framework through integration of learnings during the project lifecycle.

Enterprise-Wide Results

Building on the success of the Air Shuttle project, we developed reusable libraries, which have reduced overall mobile application development by up to 25 percent (for example, from eight weeks to six weeks). The libraries enable this savings in the following areas:

- **Design.** Layouts, graphics, and branding requirements are up-to-date and approved, freeing developers from time spent on design issues and providing users with a consistent UX.
- **Code translation time.** Developers can focus on business logic and connecting application features to the back end, which is already in place.
- **Security and legal audits.** With common corporate logic and the bulk of the security code already inspected and approved, a full security or legal audit for every new application is unnecessary.

We were able to port components of these reusable libraries to the Web/HTML5 and hybrid mechanisms so that overall mobile application development time is accelerated, no matter which delivery mechanism is chosen. See "Key Learning #2: Develop Reusable Libraries".

We also applied lessons learned to other scenarios, specifically integrating with enterprise services. From human factors engineering grew a standard look and feel and secure components for communications with the corporate domain. In addition, we applied learnings to web applications, Intel Trusted Application Portal, and hybrid applications.

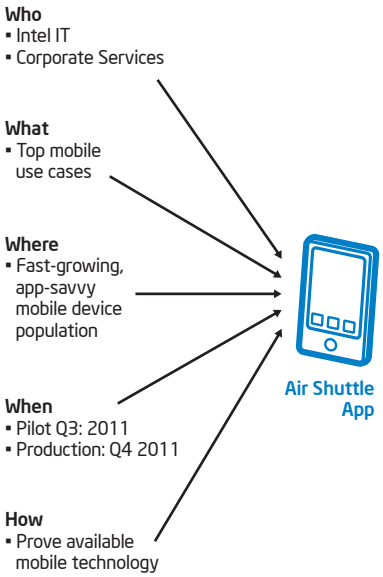

Figure 2. We partnered with Corporate Services and identified the scope and purpose of the Air Shuttle project.

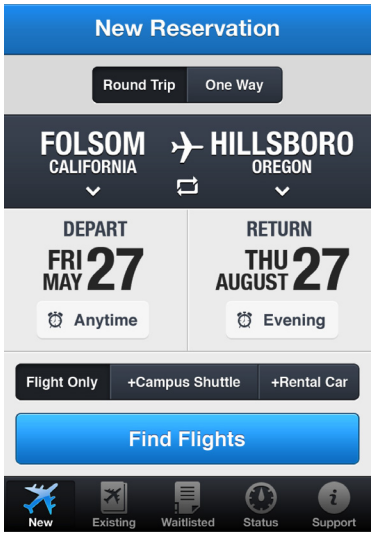

Figure 3. The mobile version of the Air Shuttle application's user interface is designed for screens that are smaller than those on laptops.

Collaborative Project Sets High Standard for Mobile Application Development

The Air Shuttle project was notable for the level of collaboration achieved between numerous business groups. The project was chartered by a partnership between IT Client and Collaboration Engineering and Corporate Services. The IT Capability Segment Teams solicited employee feedback on which Air Shuttle features to mobilize. The project team also partnered effectively with other teams in networks, secure connectivity, and security to deliver the first approved path between the mobile device and the Intel network.

APPLYING KEY LEARNINGS TO OTHER DELIVERY MECHANISMS

Using the native mechanism to develop our first mobile applications made sense at the time, because HTML5 was still maturing. The native mechanism is still part of our core mobile application strategy, but we are moving toward the hybrid and Web/HTML5 mechanisms for developing enterprise mobile applications. From our experiences with native mobile application development, we learned key lessons that we have applied to the hybrid and Web/HTML5 delivery mechanisms to improve the overall mobile application development framework.

Key Learning #1: Break Applications into Components

From our early efforts developing native applications, we realized it was critical to create once and then stage for reuse. This led to a new philosophy for developing applications: Instead of developing the whole application, we break it down into components, enabling developers to reuse chunks of code for functions common to all applications. By separating connectivity, security, and manageability code into reusable libraries, we enabled development teams to focus on the business logic and unique features of their applications.

We separated application logic into representational state transfer (RESTful) services, where the client application maintains the session state and the application logic. Decomposing the business logic into a stateless layered system allowed us to enable business functionality through the corporate gateway and firewall protections, while enabling load balancing for performance purposes.

This componentized approach has become our corporate standard for mobile application delivery.

Key Learning #2: Develop Reusable Libraries

We first developed reusable libraries for native applications, providing common application utility across OSs and devices. Libraries provide a starting point for developers as well as all the building blocks they need, while insulating them from enterprise back-end services such as security and authentication. This frees developers to focus on application functionality and business logic.

We then took most of these native building blocks and recreated functional equivalents for the Web/HTML5 and hybrid development environments. As we continue to develop mobile applications, our lessons learned in each development environment are multi-directional. We take the best functionality in each library and apply it to the other libraries.

Our reusable code libraries include these features:

- **Enterprise signing and release mechanisms to maximize cross-app permissions.** Signing allows for software authenticity and validation that the software hasn't been tampered with. The standardized release mechanisms allow for consolidated distribution of applications across devices and platforms.
- **Streamlined security and connectivity mechanisms.** By creating a standardized template of connectivity, we can quickly and easily deploy additional applications without needing to recreate security and connectivity mechanisms each time.
- **Security, legal, privacy, and geo-location guidelines.** Because many devices are employee-owned, we had to negotiate between IT control and involvement and the employee's control and ownership of the device. This required a security review and approval on implementations. We partnered with the Privacy team to review and provide guidance on what kind of data could be used and stored and how it should be disposed of in the future.

One key consideration in establishing the guidelines was the employee's location and how this information could be used. Many of our applications use location services to speed up data entry (for example, our conference room booking application can preselect a building based on the employee's location). To support our privacy policy, users can decline the use of their location and manually select their current location.

- **Centralized code storage, access, and updates.** By consolidating BKMs into code frameworks, we can ensure code reuse by future projects. We also have a centralized location that stores updated core code components instead of having them spread across various code bases.
- **Improved UX through common layouts and look and feel.** By using a common look and feel, developers provide users with a familiar interface regardless of the platform or OS they are running.

Key Learning #3: Explore Creative Solutions

An important lesson that we learned while developing our first native applications was to explore technical solutions from a different angle. We had to be creative to learn what could work on a mobile device and what was secure. Additionally, when mobilizing traditional desktop applications, it was important to consider the applications' key functionality when users were away from their primary computing system, while also taking advantage of mobile device sensors and features.

For example, in our first foray into mobile application development, we ran into the challenge of enabling VPN on a mobile device. Due to Intel's increased security guidelines, we could not allow full VPN access from mobile devices, particularly as many of the devices being used were BYO. To find a workable solution, we put additional controls into the back end to analyze behavior and identify what the normal behavior was. We decided to use a whitelisting approach where identified applications are explicitly allowed, enabling only the desired services. Other applications are not allowed enterprise or service access when connecting.

Key Learning #4: Choose the Best Delivery Mechanism for the Use Case

We have learned that no single solution fits every use case; there are multiple ways to solve a problem, and we consider the pros and cons of each delivery mechanism for each use case. Our primary strategy is to use the Web/HTML5 or hybrid mechanism for mobile application development, with the native mechanism being the exception. However, as shown in the case of the Air Shuttle project, the native mechanism was determined to be the best fit for the business use case after we weighed the pros and cons of each mechanism. At that time, native application development using VPN offered the best UX and streamlined connectivity for users when compared to the available mobile web portal technology and relatively immature state of the HTML5 standard for cross-browser development.

Instead of relying only on a single solution, we instead explore and invest in HTML5, hybrid, and native solutions as appropriate.

Key Learning #5: Integrate with Third-Party Suppliers

Though our third-party suppliers do not use our internal libraries, they are required to traverse the same connectivity path and integration through Intel's firewall. From our experiences with mobile application development, we are able to educate suppliers on the best approach for how they can integrate their own libraries with ours and how they can integrate with our back-end services while traversing the firewall.

Applying our lessons learned, we can hold our suppliers accountable to the same methods for application connectivity and security audits that we use.

Key Learning #6: Collaborate with Other Business Groups

Various business groups within Intel IT were starting mobile application development without working together or sharing BKMs. These groups experienced many of the same challenges without collaborating on the best technical solutions. The Air Shuttle project taught us the value of collaboration with diverse groups, both within and outside of IT. Together, project stakeholders defined the scope and purpose of the project and involved the key business groups throughout the duration of the project. This collaboration enabled us to successfully develop and deploy the Air Shuttle mobile application and set a precedent for collaboration in the future.

CONCLUSION

Our mobile application development framework supports the building, deploying, and sustaining of mobile applications on various form factors. The delivery mechanisms that we use to develop mobile applications are native, Web/HTML5, and hybrid. To validate the mobile application development framework, we focused on the native mechanism first, due to its maturity, richness of UX, and streamlined connectivity, especially when compared to the relatively immature state of the HTML5 standard at that time. Eventually, the lessons we learned while creating those early native applications informed our development efforts with mobile applications of all kinds.

Early native mobile development efforts revealed many technical challenges, such as slow performance on mobile devices, lack of full VPN and LAN access, and significant changes to the overall developer environment and ecosystem. The Air Shuttle project offered a better approach to native mobile application development. That project, which was the first to test the mobile application development framework, led to key learnings and BKMs that helped refine the framework.

As we shift our focus from native functionality and features to newer emerging and maturing technologies, these learnings have transferred over as well. For all delivery mechanisms, we practice componentizing mobile applications and connecting features to back-end services. For the native delivery mechanism, we developed reusable libraries and ported them to the Web/HTML5 and hybrid delivery mechanisms. Shifting our focus away from desktop and notebook applications, we explore creative solutions for meeting mobile application needs. We consider the best mechanism for meeting the business needs of the mobile application while weighing the pros and cons of each at the time. Applying our lessons learned in connectivity and security, we educate our third-party suppliers on how to connect to our back-end services. We share challenges and learnings between business groups in order to streamline development effort for all parties. By having one strategic direction and common resources, we are able to eliminate duplicate efforts and improve the process overall.

RELATED INFORMATION

Visit www.intel.com/it to find content on related topics:

- "Best Practices for Enabling Employee-owned Smart Phones in the Enterprise"
- "Building a Mobile Application Development Framework"
- "Implementing a Cross-Platform Enterprise Mobile Application Framework"

ACRONYMS

BKM	best-known method
BYO	bring your own
MEAP	mobile enterprise application platform
REST	representational state transfer
UX	user experience

For more information on Intel IT best practices, visit www.intel.com/IT.

Printed in USA ♻ Please Recycle 0114/JGLU/KC/PDF 329831-001US

Look Inside.™