Cross-platform mobile development

March 2011

Authors

Gustavo Hartmann
Geoff Stead
Asi DeGani

Address for correspondence

Tribal
Lincoln House, The Paddocks
347 Cherry Hinton Road
Cambridge
CB1 8DH
United Kingdom
m-learning@tribalgroup.com

Web addresses

www.mole-project.net
www.triballabs.net
www.m-learning.org

License

This work relates to a Department of the Navy Grant N62909-11-1-1032 issued by the Office of Naval Research Global (ONRG) and funded by the U.S. Army Medical Research and Material Command, Telemedicine and Advanced Technology Research Center (TATRC) entitled Medical Mobile Development Project.

1 Introduction

Developing effective learning solutions for mobile delivery involves multiple disciplines and a wide range of technologies. Learning applications should be engaging, deliver a consistently meaningful user experience across different devices and work seamlessly both online and offline. This may sound simple in theory, but is in fact highly complex, technically, due to a range of factors such as the highly fragmented mobile technology landscape, rapidly evolving standards, limitations imposed by the mobile device itself (screen size, input methods, display capabilities, etc.) and also constraints of the mobile network such as high latency and low bandwidth.

As mobile applications become increasingly popular and the technology moves into mainstream, the demand for sustainable practices and tools for building and supporting ongoing development becomes apparent. Although there has been lot of effort within the community—with significant work from players including the W3C Mobile group and the Open Mobile Alliance to promote best practices and mobile standards—consistent guidelines and frameworks to address common problems like the delivery of cross-platform content that works seamlessly on any device are still lacking.

M-learning providers want apps to deliver consistent user experience and behave similarly across different devices and platforms. We also want to make sure that we can achieve this in a cost-efficient manner which minimizes the need for content adaptation and also in a way that supports the evolution of mobile technology. As device capabilities such as screen size, input types and display capabilities can vary dramatically from vendor to vendor, the challenge of writing cross-device content can be significant.

As each vendor implements its own application development stack, achieving cross-platform and cross-device consistency is a non-trivial task. Fortunately as the web becomes ubiquitous and its technologies evolve, with more and more mobile browsers implementing new standards like HTML 5, CSS 3 and JavaScript, web applications are rapidly becoming an attractive and cost-efficient way of developing mobile applications that can rival native apps in terms of rich user experience and access to advanced capabilities like storage and geo-location.

Developing cross-platform mobile apps can be achieved in several ways. This document attempts to look at the state of existing frameworks that facilitate and enable it. The research focuses mainly on open source frameworks although some commercial tools are also listed for completeness.

2 Approaches to cross-platform mobile development

As hinted in the introduction, the challenges for content providers and software engineers when implementing mobile apps that can run seamlessly without modifications on different platforms are great. Fortunately, tools and frameworks have flourished and evolved over the last couple of years and can go a long way to help achieve this without compromising quality, user experience and performance.

One possible solution to solve the cross-platform problem is to use a technique called cross-compilation: a cross-compiler separates the build environment from the target environment, effectively decoupling a source from its target. In the mobile development context it works as follows: the framework provides a platform-independent API (application programming interface) using a mainstream programming language (like JavaScript, Ruby or Java). Developers use the API to build the mobile application, including the UI, data persistence and business logic. The code is then processed by a cross-compiler that transforms it into platform-specific native app(s) targeted at the different platforms that the application will run on. The software artifact generated from this process can be deployed and executed natively on the device.

The advantages of this technique are: performance, as the application is running natively on the device; improved user experience, since the app behaves like a regular app on the user's ecosystem; and full native access to a range of device specific capabilities like integrated camera, sensors, etc. The big disadvantage is complexity since cross-compilers can be difficult to write and need to be kept consistent with the fragmented mobile platforms and operating systems available.

A variation of this technique applies a virtual machine (VM) to abstract the target platform details from the application's running code. A VM is a software implementation of a machine (i.e. a mobile device) that executes programs like a real physical machine. In this scenario, the framework provides not only the API but also the runtime environment within which the application will run on. This runtime executes on the mobile device and enables the interoperability between the device's operating system and the mobile application.

This approach shares some advantages with the cross-compilation technique, although applications tend to run a bit slower in such environments due to the runtime interpretation latency introduced when the VM is translating data and instructions to and from the underlying host platform. The main advantage over cross-compilation is portability since VMs are easier to maintain and more flexible to extend when new features are added to the device and need to be supported by the API.

Another increasingly popular approach is to build the app as a mobile web application that will run on the user's mobile browser. This involves using standard web technologies like HTML, CSS and JavaScript to build the application and make it look and behave like a native app. This is possible due to the advanced capabilities of HTML 5 and CSS 3, including embedded SQL databases, local storage, animations, canvas, web sockets and video playback.

Although HTML 5 is still a young technology (the standard is yet to be finalized) and mobile browsers may implement it differently, its increasing popularity in rendering engines like the

WebKit which powers the iPhone and Android mobile browsers are making web apps look and behave more and more like native apps.

For certain classes of application this approach may be appealing as it's cheap and potentially covers a wide range of platforms with few changes. This includes common business applications like news readers, e-books, mobile banking, social interaction and e-mail. However it might not be suited for highly interactive, CPU-intensive and visually rich applications like games, augmented reality browsers and videoconferencing.

The web app runs either on a standalone mobile web browser (pure web) or in a browser-view embedded into a native app (hybrid web). In this hybrid model, the web app runs inside a thin wrapper native app which provides a bridge to the device's operational system and services. The web application is cached locally on the device on installation, removing the need for an active data connection and improving its speed and responsiveness. The communication between the web app and the native app normally happens over JavaScript via custom built APIs.

This technique tries to bring the best of both worlds into one single integrated solution: flexibility of web apps with speed and feature richness of native apps. When properly developed this tends to bring good results although since even mobile browsers are not implemented equally across devices this can also lead to inconsistencies which again can make the user experience poorer and limit the capabilities of the app.

The web approach brings some advantages like simplified deployment and immediate availability since most modern phones come with a browser installed and to run the app the user just needs the URL and an active data connection. The big drawback would be a poorer user experience since sometimes it is impossible to emulate a native UI inside a web browser using only standard web elements. Also access to advanced device capabilities like contacts, storage and sensors is normally restricted although still possible.

Finally, leveraging web technologies, vendors have created another way for mobile web sites to run like native installed applications. The approach has many different names, which can be confusing, but is normally referred to as 'mobile widgets,' and it has existed for some time now on a variety of mobile platforms. The widget concept was introduced long before the mobile app and app store revolution and can be seen as a first stab at delivering small nuggets of functionality in a lightweight and intuitive way to the end user. Popular mobile widget platforms include Symbian/Nokia Web RunTime engine, Sony Ericsson's Xperia widget engine, BlackBerry's Widget SDK and Samsung's TouchWiz widgets.

A widget is an interactive tool that provides single-purpose service to the user such as showing the latest news, current weather, date and time, calendar, dictionary, map, calculator or even a language translator [Wikipedia]. On mobile phones, widgets normally appear on the home screen or virtual desktop.

A widget runs on a widget engine which is responsible for providing basic infrastructure and accessing the device resources. Widget platforms were created initially for desktop computers to provide quick access to regular used features. Desktop widgets were pioneered by the popular Apple's Dashboard on the Mac OS followed by Windows Gadgets and Google Desktop.

Mobile widgets are small apps normally written using standard HTML, JavaScript and CSS. The usage of web technologies is invisible to the user, and the application can work just like any other software installed on the device. The platforms normally provide JavaScript APIs so widgets can access device capabilities such as camera, contacts and storage like a regular native app. This is very similar to the hybrid approach described earlier, the only difference being the packaging and access to phone capabilities since normally widgets API's are richer than bespoke cross-platform libraries based on HTML 5.

Although standards were created to help promote and standardize the widget landscape like W3C's Widget 1.0 (which works also with desktop and web widgets) and Open Mobile Terminal Platform's BONDI, they are still not widely adopted by vendors, which creates a similar fragmentation problem on the mobile app development world.

Fortunately, since most widget engines leverage JavaScript, which is a dynamic language, it is possible to reuse almost all the code, creating a multi-platform widget engine. We need to use a JavaScript wrapper API to access the core features for all the platforms, create all the possible configuration files, and create every package dynamically, changing the extension and serving it using the right MIME type.

Each of the full-stack frameworks investigated in this research use at least one of the approaches mentioned here. Some even combine them to try to improve consistency and provide a richer toolbox for the developer. Unfortunately, as in any software development effort, there's no straight answer and the solution will be driven by the type of content or service one is trying to build. In the context of mobile learning, one has to balance feature richness and broad reach to be able to select any given method.

3 Types of framework

There are a variety of frameworks and tools out there and it is useful to make a clear distinction of what they offer and their capabilities so that we can make better judgment and compare them appropriately. For the purpose of this research we divided the frameworks into the following high-level groupings:

Library: Small, self-contained toolkit that offers very specific functionality to the user. Normally used in conjunction with other libraries and tools to make up the full mobile app. Examples include UI widgets and 3D graphics libraries.

Framework: A set of libraries, software components and architecture guidelines that provides the developer with a comprehensive toolkit to build a complete mobile application, from top to bottom. Commonly called full-stack development frameworks; normally more complex to use than a single library.

Platform: A set of frameworks, tools and services that not only allow the user to build a complete mobile application but also to configure, package and distribute it to app stores or the cloud. Platforms normally include some sort of integrated development environment to ease app construction, comprehensive documentation, support and automation tools.

Product/Service: Offers a specific functionality or service ready to be used and integrated into a mobile solution. A product is built using a combination of libraries, frameworks and/or platforms but these are not normally visible to the end user.

It is also worth making a distinction between user types in the context of this research. When we say ‘user’ we refer to the developer, software engineer or architect building the mobile application. This implies that he/she has technical knowledge. When we say ‘end user’, we refer to the final consumer of the service or functionality the app or solution provides. End users frequently have little technical expertise.

4 Framework summary

The table below summarizes the most significant players as at the end of 2010 when this research was conducted:

Framework	URL	License	Type
Rhodes	http://rhomobile.com/products/rhodes/	Open Source	Platform
Phonegap	http://www.phonegap.com	Open Source	Framework
FeedHenry	http://developer.feedhenry.com/	Commercial	Platform
Appcelerator	http://www.appcelerator.com/	Open Source	Platform
Grapple	http://www.grapplemobile.com/	Commercial	Framework
MotherApp	http://www.motherapp.com/	Commercial	Framework
Corona	http://www.anscamobile.com/corona/	Commercial	Product
Sencha Touch	http://www.sencha.com/products/touch/	OS/Commercial	Library
MoSync	http://www.mosync.com/	Open Source	Platform
Resco	http://www.resco.net/	Commercial	Platform
CouchOne	http://www.couchone.com/products	Commercial	Platform
MobileIron	http://mobileiron.com/	Commercial	Platform
WidgetPad	http://widgetpad.com	Open Source	Platform
AML	http://www.amlcode.com	Open Source	Framework
Jo	http://joapp.com	Open Source	Library
xui	http://xuijs.com	Open Source	Library
JQuery Mobile	http://jquerymobile.com	Open Source	LIbrary
JQTouch	http://jqtouch.com	Open Source	Library
QT	http://qt.nokia.com/products/qt-for-mobile-platforms/	Open Source	Framework
QuickConnectFamily	http://www.quickconnectfamily.org/	Open Source	Framework
Bedrock	http://www.metismo.com	Commercial	Platform
WebApp.net	http://webapp-net.com/	Open Source	Framework

5 Framework Review

From the beginning of 2009 there has been an explosion of new frameworks and tools being created to address mobile development, as shown in the table in section 4 (which still doesn't include everything out there). Big players and start-ups alike realized the potential and fast-growing market for mobiles, and are trying hard to come up with the right tool set to make development easier and faster.

As we can't possibly review all existing frameworks and platforms, we have decided to include the ones which we feel:

- are most relevant to m-learning (and also general mobile development)
- provide a complete toolbox for developing end-to-end cross-platform mobile apps
- have a mature code base and a large community behind it
- provide decent documentation and support
- are free and open-source.

We describe below each framework in a bit more detail and explore their respective characteristics. Later we will put them in the context of m-learning and suggest a preferred approach based on the different requirements of each m-learning type.

5.1 Rhodes

Overview: The Rhodes framework from Rhomobile has been around since 2008 and provides a comprehensive platform to support the creation of cross-platform mobile apps. It achieves this by providing a runtime environment that executes on the device wrapped around a native app. This runtime VM, which is ported to the different platforms, abstracts the communication between the mobile app and the device.

In Rhodes, Ruby (the open source programming language) is used for building the business logic for the mobile app following a best practice Model-View-Controller development pattern. Another interesting aspect of the framework is the object-relational mapper component called Rhom which enables database-independent data persistence using highly scalable key-value pair noSQL storage systems.

The UI is constructed using HTML, JavaScript and CSS. The framework provides a templating language similar to Ruby on Rails views which makes it quick and easy to develop portable UIs without compromising quality. To provide animations on WebKit-based mobile browsers it extends JQTouch, a mobile JavaScript micro-library that provides native look-and-feel to mobile web apps. It also integrates well if other JavaScript UI libraries like iWebKit, xui and Sencha Touch.

Rhodes is part of a wider suite of tools for mobile development which include RhoSync for synchronization and back-end integration, RhoHub for app development and deployment and RhoGallery for hosted cloud management.

Technical architecture: Cross-compilation technique using virtual machine. Single source codebase written in Ruby and HTML/JS/CSS running on a Ruby interpreter on the device. The

Rhodes runtime (which contains the interpreter) is wrapped around a native app and provides cross-platform common abstractions for accessing device capabilities. On platforms where the primary development language is Java, such as BlackBerry, Rhodes applications are cross-compiled into Java bytecode that are then executed natively. On iPhone, Android, Windows Mobile and Symbian platforms, Rhodes applications are compiled into Ruby bytecode.

Supported platforms: iOS, Android, Blackberry, Windows Mobile 6.5, Symbian

Strengths: Ruby code helps to structure and control business logic using the built in Model-View-Controller and Object Relational Mapper design patterns. Supports a broad range of mobile platforms.

Weaknesses: Updating HTML/JavaScript code needs a complete rebuild. Need to know Ruby well to do anything a bit more sophisticated, which is not as popular as other programming languages like JavaScript, Java or PHP. Doesn't generate source code only native package which can restrict any further tweaking of the app.

Version reviewed: 2.2.6

Access to Device Capabilities

Capability	iPhone	Windows M	BlackBerry	Symbian	Android	Palm
Geo-location	Yes	Yes	Yes	Yes	Yes	N/A
PIM contacts	Yes	Yes	Yes	Yes	Yes	N/A
Camera	Yes	Yes	Yes	Yes	Yes	N/A
Native menu/Tab bar	Yes	2.0	Yes	2.1	Yes	N/A
Barcode	2.1	2.1	2.1	2.1	2.1	N/A
Audio/video capture	3.0	3.0	3.0	3.0	3.0	N/A
Bluetooth	2.2	2.2	2.2	2.1	2.2	N/A
Push/SMS	Yes	2.0	Yes	2.1	2.0	N/A
Calendar	2.2	2.2	2.2	2.2	2.2	N/A
Screen rotation	2.1	2.5	2.0	2.1	2.1	N/A
Native maps	1.4	2.3	1.4	2.1	1.5	N/A
Ringtones	2.5	1.5	1.5	N/A	1.5	N/A
Storage	2.0	2.0	2.0	2.0	2.0	N/A

License: MIT, permits reuse within proprietary software on the condition that the license is distributed with that software.

5.2 Phonegap

Overview: The Phonegap open source framework from Nitobi created in early 2008 provides a decent toolbox for building native mobile applications using only HTML, JavaScript and CSS. It's quite popular among users mainly because of its flexibility, straightforward architecture and ease of

use. You basically need to drop the libraries in the right place and start coding with a familiar web stack to produce a functional app quickly.

It follows the hybrid web model discussed earlier by providing a set of native app wrappers for all major mobile platforms with an embedded browser (also know as WebView) which renders the UI and supports the interaction between the web app and the device. The core part of the framework is its device-independent JavaScript API that provides rich functionality like storage, geo-location, sensor interactivity, etc without coupling the code to the specifics of the underlying platform.

Phonegap is probably best suited when you have an existing web application you want to convert/port to a mobile environment. Since most code would already be in a web format anyway, the task of converting vanilla web to mobile web in most cases shouldn't be too labor-intensive (obviously depending on how the web app was built and its nature).

Although the API provides a useful and feature-rich set of capabilities it falls short when it comes to UI and making your app look and feel like a native citizen. The creators focused more on the meaty features (and making them consistently available across platforms) and left the users with the task of styling the markup to mimic a native app. Fortunately there are loads of open source libraries specialized in exactly this. Therefore it's not unusual to see Phonegap being used in conjunction with other libraries like XUI and Sencha Touch.

Another important aspect of the framework is that it imposes little structure and/or guidelines on how to best develop applications with it. This means users are free to architect their solutions in a way that best suits their needs. This can be a blessing for experienced developers but can create confusion and promote bad designs within novice users. This minimalist approach is possibly the reason why the framework can cover a wider range of target platforms than any other open source tools reviewed in this research.

Technical architecture: Web approach using hybrid model. Single source codebase written in HTML, JavaScript and CSS running on a mobile browser embedded on a thin native app wrapper which exists for every supported target platform. Access to device's capabilities happens thought a device-independent JavaScript API which talks to the OS proprietary API's also via JavaScript.

Supported platforms: iOS, Android, Blackberry, Windows Mobile 6.5, Symbian, Palm

Strengths: All native wrapper source code is provided so it can be customized further. Simple 'drop-in libraries' concept makes it easier to develop. Broad range of platforms supported. Apps built purely in HTML, JavaScript and CSS makes lowers the barrier of adoption for web developers.

Weaknesses: Must assume that normal capabilities of a web-based application are available. Recommended as a contender for applications which are heavily web dependent. Lack of support for native UI components, design patterns and dev tools.

Access to Device Capabilities:

Capability	iPhone	Windows M	BlackBerry	Symbian	Android	Palm
Geo-location	Yes	Yes	Yes	Yes	Yes	Yes
PIM contacts	Yes	Yes	BB OS 5/6 only	Yes	Partially	No
Camera	Yes	No	BB OS 5/6 only	Yes	Yes	No
Native menu/Tab bar	No	No	No	No	No	No
Barcode	No	No	No	No	No	No
Audio/video capture	Partially	Partially	No	No	Yes	No
Bluetooth	No	No	No	No	No	No
Push/SMS	No	No	No	No	No	No
Calendar	No	No	No	No	No	No
Screen rotation	Yes	Yes	BB OS 5/6 only	Yes	Yes	Yes
Native maps	No	No	No	No	No	No
Ringtones	No	No	No	No	No	No
Storage	Yes	No	BB OS 5/6 only	No	Partially	Yes

Version reviewed: 0.9.3

License: MIT, permits reuse within proprietary software on the condition that the license is distributed with that software.

5.3 Appcelerator Titanium

Overview: Titanium from Appcelerator Inc. has been around since December 2008 and provides a rich set of tools not only for cross-platform mobile phone development but also desktop and tablet application development. Its excellent documentation and online resources coped with its standalone app life cycle management environment makes even the inexperienced user productive very quickly.

Titanium's approach to cross-platform is based on the cross-compilation technique: a platform-independent JavaScript API which is compiled to the different target platforms as required. This compilation process happens in three steps: pre-compilation, front-end compilation and platform and package compilation.

The pre-compilation takes the app's JavaScript code, optimizes it (reduces whitespace, reduces the size of symbols, etc.) and then creates a dependency hierarchy of all the Titanium APIs used. The front-end compilation step generates the appropriate platform-specific native code, native

project (if necessary) and build any specific code that is necessary to compile Titanium for a given platform compiler. The platform compiler and packager step effectively compiles the code to native executable using platform specific tools and packages files for running either on the native simulator, native device for testing or for final packaging for distribution.

One key difference between Titanium's approach to other frameworks reviewed in this research is that although it uses JavaScript as the main language for development, it doesn't use an browser engine to render the user interface on the mobile phone device (although it does for desktop apps). Its API provides abstractions to most UI elements that are converted to true native UI elements when deployed to the phone. This reveals a much fluid and richer user experience than the browser-based techniques.

Technical architecture: Cross-compilation technique. Single source codebase written in JavaScript compiled into native code and packaged for the different target platforms.

Supported platforms: iOS, Android, BlackBerry (coming soon)

Strengths: Native code output so very quick and fluid on phone. Easy setup and start-up for developers. Excellent documentation and examples. Strong community forums to find out answers. Intuitive app management environment. Potential support for desktop and tablet development.

Weaknesses: Potentially restrictive API's. Small set of phones currently supported. Tries to solve too many problems in one single shot (i.e. supporting phones, tablets and desktops)

Access to Device Capabilities:

Capability	iPhone	Windows M	BlackBerry	Symbian	Android	Palm
Geo-location	Yes	N/A	N/A	N/A	Yes	N/A
PIM Contacts	Yes	N/A	N/A	N/A	Partially	N/A
Camera	Yes	N/A	N/A	N/A	Yes	N/A
Native menu/Tab bar	Yes	N/A	N/A	N/A	Yes	N/A
Barcode	No	N/A	N/A	N/A	No	N/A
Audio/video capture	Yes	N/A	N/A	N/A	Yes	N/A
Bluetooth	No	N/A	N/A	N/A	No	N/A
Push/SMS	Partially	N/A	N/A	N/A	Partially	N/A
Calendar	No	N/A	N/A	N/A	Yes	N/A
Screen rotation	Yes	N/A	N/A	N/A	Yes	N/A
Native maps	Yes	N/A	N/A	N/A	Yes	N/A
Ringtones	No	N/A	N/A	N/A	No	N/A
Storage	Yes	N/A	N/A	N/A	Yes	N/A

Version reviewed: 1.5.1

License: Apache License 2.0, permits reuse within proprietary software but requires preservation of the copyright notice and disclaimer agreements. Commercial licenses are available for enterprise customers wishing premium support and additional services.

6 Making the right choices for m-learning development

With the accelerated growth and ever increasing capabilities of mobile devices, it is an exciting world for m-learning development. Educators and learning technologists are now capable of innovating in ways that were never imagined just a decade ago when m-learning first became popular and its possibilities identified as a disruptive medium to improve learning and engage end users anywhere at anytime.

As the mobile software industry evolves and matures, new affordable frameworks become available to leverage mobile development. And this is happening at a fast pace as indicated in previous chapters. The proliferation of all kinds of libraries and platforms (either commercial or open source) can cause confusion and lead to wrong decisions being made at the start of an m-learning project. One not only needs to reach the broadest audience (and support a wide range of devices) but also to choose carefully the technical approach based on the different types of m-learning activities that will be developed.

This section will briefly describe the different (and more commonly used) types of m-learning and suggest the approach which might be more suited for the job from a technical perspective. Although we will recommend one technique/tool, we appreciate the fact that sometimes the ideal solution may involve mixing different tools and recipes. So this is not meant to be a definitive analysis of the subject, but rather a guide based on experience which we are hoping could contribute to increase the overall awareness on what's possible and feasible when doing m-learning.

6.1 Types of m-learning

This taxonomy is based on previous work from the MoLE project, where it is discussed in more depth. We are listing them here together with a brief description in order to familiarize the more technical reader with current m-learning terminology:

mVLE-based learning: A mobile extension to virtual learning environments (such as Moodle, BlackBoard, Atlas Pro) allowing users to participate in activities that happen in the non-mobile version of the environment.

Content delivery: A traditional linear content, whether casting ('pod' or 'vod'), e-books, apps or full SCORM packaged courses. The learner makes a conscious decision to 'learn.'

Record of achievement: Mobile devices act as a quick way to record and demonstrate achievement for e-portfolio (or talent management) systems. Smartphones offer the ability to record evidence anywhere and anytime with little preparation.

Just-in-time training: (aka 'performance support') A scenario where a learner requires immediate information for the completion of a task. The mobile device can be used to obtain the information through a corporate learning portal.

Social learning: Where the learning is a product of peer interaction. This is happening in both educational and corporate environments and is sometimes referred to as 'collaborative' learning.

Enhanced reality: Not a learning type per se, this type of technology is used in a number of learning contexts to deliver geographically relevant content. The term relates to the enhancement of the users' visual environment by overlaying pictures or text over the live image captured by the device's camera.

Learning support: Where the phone is used to facilitate learning from another source. An example would be a dictionary running on the phone while the learner is reading from a book.

Experience-based learning: A mode of learning that relies on the devices' functionalities to enhance the experience (GPS, camera, etc.). A class group on a field trip collecting images of different architectural styles is a good example.

Game-based learning: A mode of learning that is very common on the PC and is made possible by the increasing performance of the mobile platforms. The games need to be designed specifically for the small screen if they are to succeed.

6.2 Choosing a technical approach

Reaching the broadest possible audience is a common desired goal when building m-learning products and services and it involves building apps that can run on a wide range of mobile devices which inevitably have distinct capabilities. This is not an easy task and can be successfully achieved by not only choosing the right cross-platform technique but also by recognizing that certain types of requirements can't be done in a device-independent manner without sacrificing the user experience or duplicating efforts.

With that in mind, we think cross-platform m-learning can be technically achieved with decent results by leveraging the mobile web and its related set of standard technologies, which include HTML, JavaScript and Cascading Style Sheets. The learning scenarios where this technique can be applied with a good degree of success are mainly mobile VLEs, content delivery of most kinds, record of achievement, just-in-time training, social learning, learning support and some types of experience-based learning.

Using the web to deliver m-learning in these cases, where the web browser provides most of the plumbing, was only made viable in the last few years by the rapid evolution of the mobile browser. Frameworks like PhoneGap or Rhodes explore this extensively providing access to advanced phone capabilities that were until recently the territory of native applications. It also enables a high level of code reuse since the bulk of the application logic can be safely reused across platforms as already explored on previous chapters.

However, the web approach has limitations and at the moment is only suitable in learning applications that do not require highly interactive graphics and/or CPU intensive processing power. In such cases, writing native applications using the full power offered by modern mobile operating systems is the way to go, even though it might mean having to port code to the different target platforms. In this case, portability is sacrificed for the sake of a richer and truly meaningful learning experience. Because such applications require very specific and highly sophisticated infra-structure most of times they are only available to a few newer platforms and do not aim to reach the broad crowd out there still on old kit.

These applications are typically represented by game-based learning (aka 'edutainment'), enhanced reality (or augmented reality) and some types of experience-based learning. In the realm of games, truly immersive and engaging 3D worlds are a necessity. The use of games for learning is popular and the increasing graphic capabilities of modern devices made richer gaming experiences possible and readily available to the mobile learner. Augmented reality is another area where intensive CPU processing is required to be able to calculate and draw 3D objects in real-time on top of the live camera view.

Even though frameworks like Appcelerator Titanium are capable of generating native UI elements, they still only allow for very basic user interactivity and lack more advanced techniques like 3D rendering and multi-threaded programming. This is an area where native apps are king and it will probably still take a few years before these capabilities are widely available and supported by mobile browser developers.

The table below summarizes the different m-learning types and the suggested technical development approach to be taken:

M-learning type	Technical approach
mVLE-based learning	Cross-platform, web-based, hybrid model
Content delivery	Cross-platform, web-based, hybrid model
Record of achievement	Cross-platform, web-based, hybrid model
Just-in-time training	Cross-platform, web-based, hybrid model
Social learning	Cross-platform, web-based, hybrid model
Enhanced reality	Fully native, ports to different platforms
Learning support	Cross-platform, web-based, hybrid model
Experience-based learning	Fully native or Cross-platform
Game-based learning	Fully native, ports to different platforms

The important thing to keep in mind when choosing the technology stack for any given m-learning development is this: there's no single tool, framework or technique that can reach all devices on all platforms without giving up some functionality or the need for porting some bits of code. And this is a reality not only for mobile development but also for most types of current software development efforts including web, desktop and embedded.

7 Conclusions

Mobile learning is here to stay and has the potential to change the way we perceive and engage with traditional learning tools and paradigms. It can make learning widely available to people anywhere at anytime. The fast evolution of mobile platforms has enabled a fascinating world of new learning solutions which captivate and bring the learner to the center of the learning experience and not just a passive observer.

To exploit this fully, content providers, learning technologists and software engineers need to make use of the right set of development tools to be able to quickly and effectively create engaging mobile learning solutions. The recent boom observed in the mobile development landscape has partially enabled this although the options are too many often causing confusion and potentially creating even more challenges to organizations willing to embark into m-learning with little or no previous experience.

This research examined novel technical approaches for cross-platform mobile development, especially looking for ways of building, deploying and maintaining mobile applications which run with little or no modification on all major mobile phone platforms, including iOS, Android, BlackBerry, Windows Phone, Symbian and Palm.

The current state of cross-platform development techniques falls into two broad approaches: cross-compilation and mobile web applications. Cross-compilation is the technique where a common source language is used to build the application, which is later compiled to the different required target platforms. The m-learning app in this scenario runs as a native app on the mobile device.

The mobile web approach basically involves the use of common web technologies including HTML, JavaScript and CSS to build the application which is then presented to the user using a mobile browser, either standalone or embedded into a native wrapper application. The m-learning app in this scenario looks and behaves like a native app even though it is powered by the mobile browser.

We have explored a few popular open source frameworks, including Rhodes Mobile, Phonegap and Appcelerator Titanium which provide support for mobile development using either approaches with a detailed analysis of their capabilities, strengths and weaknesses.

Finally we have analyzed how the different types of m-learning requirements can potentially influence the technical architecture decisions, and we have provided a few recommendations and suggestions of when and how to use one approach/tool over the other.

Mobile development and m-learning is a very hot topic at the moment with a highly dynamic and vibrant community of users and solution providers. Although experience can go a long way when choosing the right approach for m-learning development, things change so fast in this field that what was a best practice a few years ago can become irrelevant as new technology breaks through.

8 References

Allen, S., V. Graupera, L. Lundrigan, *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution*. New York, Apress, 2010.

Firtman, M., *Programming the Mobile Web.* Sebastopol, O'Reilly Media, 2010.

Fling, B., *Mobile Design and Development*. Sebastopol, O'Reilly Media, 2009.

Firtman, M., *Mobile Widgets Overview. Mobile Web Programming*. W3C. http://www.mobilexweb.com/blog/mobile-widgets-overview. 22 June 2010.

Cáceres, M., *Widget Packaging and Configuration Working Draft 5 October 2010*. W3C, http://www.w3.org/TR/widgets/, 2010.

Connors, A. and B., Sullivan, *Mobile Web Application Best Practices W3C Recommendation 14 December 2010*. W3C, http://www.w3.org/TR/mwabp/, 2010.

Nicolaou, A., *The Iterative Web App: Feature-Rich and Fast*. Google Mobile Blog, http://googlemobile.blogspot.com/2009/12/iterative-web-app-feature-rich-and-fast.html, 2009.

Paul, R., *Rhomobile unveils Ruby-based cross-platform mobile framework*. Arstechnica, http://bit.ly/3VNt, 2009.

Rowberg, J., *Comparison: App Inventor, DroidDraw, Rhomobile, PhoneGap, Appcelerator, WebView, and AML*. Application Markup Language, http://bit.ly/bzcuvK, 2010.

Pilgrim, M., *Dive into HTML 5*. http://diveintohtml5.org/.

Brooks, F., *The Mythical Man-Month: Essays on Software Engineering*. Indianapolis, Addison-Wesley Professional, 1995.

Krueger, D., *Create Offline Web Applications on Mobile Devices with HTML5*. developerWorks, http://ibm.co/cNmj6E, 2010.

Stark, J., *Building iPhone Apps with HTML, CSS, and JavaScript*. Oreilly, http://ofps.oreilly.com/titles/9780596805784/, 2009.

Grigsby, J., *Mobile Operating Systems and Browsers Are Headed in Opposite Directions.* Oreilly, http://oreil.ly/dnqXL1, 2010.

Kriesing, W., *EventNinja – A Mobile Cross Platform App*. Uxebu, http://bit.ly/afpXar, 2010.

Braga, M., *Why Not All WebKit Mobile Browsers Are Created Equal*. Tested, http://bit.ly/dLJJgO, 2010.

Hazaël-Massieux, D. and F., Daoust, *Native Apps vs mobile Web*. Don't call me DOM, http://bit.ly/cop7xD, 2010.

Rohrer, J.J., *Cross Platform Development Strategies with vendor review and PhoneGap case study*. Slideshare, http://slidesha.re/1r8dRQ.

DeGani, A., E-learning Standards for an M-learning world – informing the development of e-learning standards for the mobile web.