Hybrid Mobile Application Development Approaches

The proliferation of mobile platforms and technologies, along with the evolution of smart devices, provide plenty of opportunities for compelling applications. In addition to the advancements in mobile technologies, the user interface is becoming the paramount factor in defining the adaptability of the mobile application. The key expectations of a mobile application are multifold, and include cost effectiveness, a single code base to work across multi platforms, an outstanding UI and faster delivery cycles.

Another important factor that cannot be ignored is the powerful set of device features that changed the feature phones to smart phones/tablets. The modern applications are taking full advantages of the device features (GPS, camera...), form factors (tablets) and appealing user interaction capabilities such as swiping styles, pinch and zoom, etc. The best fit to accommodate most of the key mobile capabilities (single code base form factors, great UI) is to adopt mobile web technologies. This can be augmented by using the Hybrid approach to take full advantage of the device features in addition to the benefits listed above. The paper focuses on different mobile technologies briefly, but mainly focuses on the Hybrid approach, challenges, architecture, benefits, concerns, industry trends and market adaptability.
About the Authors

Sankar Srini

Education: Masters in Engineering, Indian Institute of Technology, New Delhi
Bachelors in Mechanical Engineering, Sri Venkateswara University, India

Summary of experience: Sankar leads the TCS Hy5 Canvas™ solution – TCS’ hybrid mobile application development platform – through all phases of the software development lifecycle. He has over 20 years of professional experience in various industry segments such as CAD/CAM, core database development (at Informix Corp), Construction, transportation industry (BNSF), Mobile technologies – Wap Gateways at Motorola, CRM industry (Lexis Nexis), and Manufacturing automation (Panasonic). His primary interest lies in building product strategies, including the technical design, application level architecture, and enterprise level architecture.

Sathish Venkatraman

Education: MS Information Technology, Bharathidasan University, Trichy, India

Summary of experience: Sathish Venkatraman is responsible for the Hybrid application middleware platform development (TCS Hy5 Canvas™) middleware solution. He has over 16 years of IT industry experience including Oracle Applications ERP implementation and support, customization and migration using Oracle Tools, and has architected solutions for complex business integrations based on the industry standards OAGIS and AIA EBOs. He also has an interest in the areas of general ledger, account payables, account receivables, asset management, inventory, purchasing and order management modules.
# Table of Contents

1. Executive Summary 4
2. Introduction 4
3. Evolution of key technologies 5
4. Analysts’ view 6
5. Vendor support 7
6. Power of Hybrid over Mobile Web 8
7. Market adoption 8
8. Challenges in Hybrid App Implementation 9
9. Architectural approaches 12
10. Recommendations 13
11. Conclusion 15
12. References 15
Executive Summary

A hybrid application, by definition, is anything derived from heterogeneous sources or a combination of technologies, approaches or elements of different kinds. With respect to mobile applications, a hybrid application leverages both native and mobile web technologies. The mix of web content and native capabilities determines the degree of hybrid capabilities of the application. It can vary from 1% to 99%. This paper mainly focuses on the need for the Hybrid approach, functional advantages and disadvantages, challenges, implementation methods, impact on operational aspects such as maintenance and upgrades.

The key focus of hybrid applications is to arrive at cross platform compatibility, with an emphasis on native features. The power of native features cannot be ignored and the Hybrid approach takes full advantage of this.

According to Gartner, if your organization is targeting three or more mobile platforms, or if you require connectivity to three or more applications, then multichannel or cross-platform tools typically yield better project results during the course of three to five years when compared to native tools. Another interesting observation was that when developing an application for two platforms, the cost is 160% greater than developing for a single platform. This paper summarizes these approaches and outlines the benefits.

Introduction

The mobile industry is evolving; whether it is platforms, devices, technologies, features, network models or exploration of application use cases, the speed of change for any one of these technologies means that businesses must give serious thought before investing in creating their own applications. To address these challenges, the industry as a whole is looking for a single code base that can work across multiple platforms and devices to develop consumer apps and enterprise apps. In addition to the above, the Bring Your Own Device (BYOD) policies that are emerging in the industry accelerate the need for a single code base on the web, with little or no footprint on the device to easily manage applications. Before taking the dive into the hybrid approach, it is worthwhile to discuss other potential solutions.

Native: In this approach, an application is developed using native OS languages to best make use of individual platform SDKs and device features to generate the best possible user experience. Unfortunately, applications developed for one platform cannot be used across platforms, meaning overall development cost will be higher if one wishes to be cross-platform compatible.

Mobile Web: This approach is based on proven, standard web technologies. It is the simplest approach to creating applications for mobile devices without much investment, as it is nothing more than accessing web pages on mobile devices. Though this option is very fiscally attractive, the user experience is very limited and has its own set of challenges limiting potential adoption, as the usage of the powerful native features must be totally ignored.

Hybrid: This is designed to take advantage of both the Native and mobile Web approaches. It benefits from the versatility of web technologies with the combination of powerful device features and SDK. It is
well suited for a broad range of applications and can still provide a good user experience. This paper further discusses the hybrid approach deeply in the following sections.

Advantages of hybrid approach:

- Faster time to market when compared to pure native app development time cycles
- Lower total cost of ownership as cross platform adaptability is easier
- Reasonably good UX experience with the modern JavaScript toolkits (gestures, swipes, tap and transitions)
- Easy maintenance, upgrades and deployments
- Access to native features

Disadvantages of hybrid approach:

- Slightly lower performance due to rendering of web pages and accessing data through multiple layers
- Limited UX experience compared to native UI
- Security concerns

**Evolution of key technologies**

Having proper support for the right technologies is essential for application development; there has been a tremendous push across the industry to ramp up investment in Hybrid Applications development, specifically in these key areas:

**HTML 5**

- Version 5.0 supports many features that aid smart phone application development
- Application caching, client storage – Google™ mail
- Many new elements and attributes `<canvas>`, `<audio>`, `<video>`
- Great alternative to flash plug-in for video because of stability and power consumption issues
- Canvas – dynamic, scriptable rendering of 2D shapes and bitmap images
- Strong support for APIs
  - Geo location
  - WebSQL and Web storage - local cache of key value pairs
  - Web Sockets
  - Drag and drop features
CSS 3.0

- Significant performance boost
- Responsive web design with media queries and form factor adjustments
- Support for progressive enhancements with adjustable web page content based on device functionality
- Opacity and shadows, which provide better visual design

JavaScript

- Great support for user interface experience (such as transitions and swipes)
- New set of object properties are defined
- New functionality to handle array objects
- Garbage collector
- Numerous emerging frameworks

Analysts’ view

Industry analysts have taken a closer look at the evolution of these technologies and arrived at several recommendations. Below are some excerpts from Forrester and Gartner analysts:

“Because of the increasing consumer adoption of HTML5-ready desktop browsers, retailers will look to raise the user experience stakes of their main ecommerce sites by introducing HTML5 versions.”¹

“Developers show growing interest in technologies that promote open web architectures. HTML5 is certainly one of these, with 60% of developers either already using it or planning to within the next two years.”²

“Mobile Web apps can, in certain scenarios and with careful attention to application programming interfaces and extensions, provide a rich user experience that does not equal native apps, but approximates them at a fraction of the development effort and with greater portability and flexibility.”³

“Embrace HTML wrapping approaches and tools as a primary strategy for delivering content and application capabilities to mobile devices in a cross-device, cross-platform, cross-browser approach.”⁴

“Although there will still be app stores by 2015, over 50% of the apps sold and/or pointed to (rather than installed) will be Web apps, not platform-specific native apps. There will still be paid apps, but more will be Web apps for which users pay. More specifically, by 2015, 60% of enterprise mobile applications and 40% of consumer mobile applications will be Web applications.”³

“Instead of investing in a series of bespoke applications, the FT has built a single web application in HTML5 that can work in the browser on a range of smart phones and tablets.”⁵
## Vendor support

Browser vendors are gearing up to support the compatibility of HTML5 features in both PC and mobile media.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Safari iOS</th>
<th>Android browsers</th>
<th>IE</th>
<th>Firefox</th>
<th>Blackberry</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>iPhone iPad</td>
<td>Phone (1-2.3.4.0)</td>
<td>Tablets 3.0</td>
<td>9.0</td>
<td>Android</td>
<td>Phone</td>
</tr>
<tr>
<td>Application cache</td>
<td>3.2</td>
<td>2.1+</td>
<td>3.0</td>
<td>No</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Web storage</td>
<td>3.2</td>
<td>2.0+</td>
<td>3.0</td>
<td>9.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Web SQL Storage</td>
<td>3.2</td>
<td>2.0+</td>
<td>3.0</td>
<td>No</td>
<td>No</td>
<td>6.0</td>
</tr>
<tr>
<td>Geolocation</td>
<td>3.2</td>
<td>2.0+</td>
<td>3.0</td>
<td>9.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Multimedia</td>
<td>3.2</td>
<td>2.3+</td>
<td>3.0</td>
<td>9.0</td>
<td>6.0</td>
<td>7.0+</td>
</tr>
<tr>
<td>Server sent events</td>
<td>4.1+</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>6.0</td>
<td>No</td>
</tr>
<tr>
<td>Web sockets</td>
<td>4.2+</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>6.0</td>
<td>6.1+</td>
</tr>
<tr>
<td>Web workers</td>
<td>5.0+</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Canvas API</td>
<td>3.2</td>
<td>1.5</td>
<td>3.0</td>
<td>9.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Touch events</td>
<td>3.2</td>
<td>2.1+</td>
<td>3.0</td>
<td>No</td>
<td>6.0</td>
<td>6.1+</td>
</tr>
<tr>
<td>CSS- basic</td>
<td>3.2</td>
<td>1.5</td>
<td>3.0</td>
<td>9.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Network information</td>
<td>No</td>
<td>2.2+</td>
<td>3.0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>File API</td>
<td>No</td>
<td>4.0+</td>
<td>3.0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Source: [http://mobilehtml5.org/](http://mobilehtml5.org/)

### Table 1: Mobile Browser Support Matrix

<table>
<thead>
<tr>
<th>Feature</th>
<th>Webkit</th>
<th>Gecko</th>
<th>Trident</th>
<th>Presto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geolocation APIs</td>
<td>5.0</td>
<td>5.0+</td>
<td>3.5+</td>
<td>9.0+</td>
</tr>
<tr>
<td>Canvas support</td>
<td>5.0 Partial</td>
<td>16.0 partial</td>
<td>3.5+</td>
<td>9.0 Partial</td>
</tr>
<tr>
<td>Offline storage</td>
<td>5.0</td>
<td>13.0+</td>
<td>3.5+</td>
<td>No</td>
</tr>
<tr>
<td>Drag &amp; drop</td>
<td>5.0</td>
<td>16.0</td>
<td>3.5+</td>
<td>9.0 Partial</td>
</tr>
<tr>
<td>Audio &amp; Video</td>
<td>3.1</td>
<td>1.0+</td>
<td>3.5+</td>
<td>9.0 Partial</td>
</tr>
<tr>
<td>Server events</td>
<td>5.0</td>
<td>16.0</td>
<td>6.0</td>
<td>No</td>
</tr>
<tr>
<td>Web sockets</td>
<td>5.0</td>
<td>5.0+</td>
<td>6.0</td>
<td>9.0 Partial</td>
</tr>
<tr>
<td>Web workers</td>
<td>5.0</td>
<td>16.0</td>
<td>3.5+</td>
<td>No</td>
</tr>
<tr>
<td>Support for JavaScript (ECMA Script 3.0)</td>
<td>5.0</td>
<td>5.0+</td>
<td>3.5+</td>
<td>5.0+</td>
</tr>
<tr>
<td>File APIs</td>
<td>5.0</td>
<td>5.0+</td>
<td>3.5+</td>
<td>9.0 Partial</td>
</tr>
<tr>
<td>Support for CSS 3.0</td>
<td>5.0</td>
<td>5.0+</td>
<td>4.0+</td>
<td>9.0</td>
</tr>
<tr>
<td>Index based API</td>
<td>5.0 Partial</td>
<td>16.0 Partial</td>
<td>4.0+ Partial</td>
<td>9.0 Partial</td>
</tr>
</tbody>
</table>


### Table 2: PC Support Matrix
**Power of Hybrid over Mobile Web**

Mobile web features can be greatly augmented by placing them in a hybrid context. Key features include:

**Database storage:** In mobile web, data is stored as plain text. There is a room for security threat (a) if data is lost (b) if the storage location is fixed and other apps can access it. With hybrid applications, data can be stored securely with encryption.

**Media:** With mobile web applications, content can be played back. With hybrid applications, data can not only be played back, but recorded using the native bridge.

**Network connections:** HTML5 supports web sockets. The Hybrid approach can handle full socket communications. Native mobile components can open a socket and can communicate with the server/other devices, just like in traditional socket communication. In HTML5, the web server can support the web sockets, though not all the browsers are supporting this feature at this time.

**Push notification:** Real-time push notifications are possible with the Hybrid approach via the use of native components. HTML5 also supports notifications via server push technology with persistent connections.

In addition to the above, the hybrid approach supports full access to device features such as the camera, motion sensors, accelerometer, file system access, contact lists and sensors.

**Market adoption**

The market adoption of HTML5 has been swift:

- The announcement of an HTML5 based OS ‘Boot to Gecko’ by Mozilla™ will be a boost to hybrid application development and is expected to enter the Brazilian market in early 2013. Devices featuring “Boot To Gecko” will be manufactured by TCL communication technology (Alcatel) and ZTE.
- Leading global network operators are backing this initiative, including Deutsche Telekom, Etisalat, Smart, Sprint, Telecom Italia, Telefónica and Telenor.
- Rich Internet Applications (RIA) applications are viewing HTML5 as an alternative to Adobe Flash and Microsoft’s Silverlight technologies.
- H.264 format videos supported by HTML5 (Consumer applications)
- Google Gears project for offline storage is superseded by HTML5
  - Web storage API (Key-Value pair)
  - Web SQL Database API (WebDB) (Example: Gmail)
  - Financial sites are using HTML5 based charting (Google Finance, FT)
- HTML5 Canvas supports dynamic rendering of 2D graphics
- One billion HTML5 phones to be sold worldwide in 2013.
Challenges in Hybrid App Implementation

In this section we will explore challenges that one encounters in hybrid application implementation and how they can be overcome.

Advanced UX capabilities

Developing for the web UI has classically been much more challenging compared to a native UI. Several JavaScript tool kit vendors stepped up to close the gap and offer reasonable support for UI features such as

- Page transitions with slide up, slide down, pop, fade, flip features
- Swipes with Tap, Tap hold, swipe left, swipe right features
- Platform specific look and feel of controls
- Charting controls support

Application caching

Traditional web applications store data in session variables and cookies which are still supported by the browser cache. HTML5 extends this support to store the data in a web database as well. The size of web databases is limited (5 MB – 10 MB) and does not offer encryption. Another drawback of using web databases is that the external synchronization process cannot access the database, as the database is tied to the browser.
Offline database support

Offline database support allows users to have all required data to make the applications operational without a network connection. SQLite, UltraLite and SQL server 2005 (Windows) are some of the choices. These databases can be very large and can have relational data. A SQL Cipher offers an encrypted database.

In addition to storing application data, the offline database can be used for data transfer as a sharing media from HTML to native portions of the application and vice versa. Applications should be able to execute SQL queries on the local database and retrieve the result sets easily and process them on the web content.

Security vulnerabilities

Security vulnerabilities are generally the same for hybrid and web applications. Most modern web browsers prevent certain vulnerabilities such as malicious scripts or cross-domain requests; however, typically hybrid applications use uiWebKit (iOS), WebView (Android); these are native controls that use WebKit engines and do not offer the same level of support as a browser does. It is important for a development platform to provide tools to mitigate security vulnerabilities such as:

- Cross site scripting (Prevention of malicious scripts)
- Cross site Request Forgery (Cookie thefts)
- SQL injections
- Secured local storage (encryption)
- Data encryption: additional encryption may be required for data transfers between web and native contents or data storages on the device database.

Performance

Performance is one of the paramount concerns for the application developers. The key reasons for performance degradation are:

- Data transfer across multiple layers of native, JavaScript libraries and webview.
- Rendering of web pages from the server. This can be avoided by embedding the web content as part of the application footprint on the local device. One of the key challenges is figuring out how to handle the changes that can occur to the web content on the server side. Active monitoring of web content can update the application footprint on the mobile device. This can eliminate the need for rebuilding and packaging processes.
- Loading of larger images. This can be improved by loading the images from the device and can be further optimized by choosing the appropriate image dynamically using device detection and form factor considerations.
- Optimized data transfers over the network.
Content adaptation/switching

Hybrid applications have the challenge of dynamically adjusting the content to the form factor of the web view window. There are several approaches one can take. The most common way of adjusting the web content is to use CSS.

The content adaptation can be done on the server, client or both.

**Server:** This paradigm relies on the device detection libraries or databases installed on the webserver (such as WURFL, DeviceAtlas) to detect the device accessing the web site and provides the device capabilities. These sets of capabilities allow the developer to fine-tune the resulting page to match the device's capabilities with a very high level of control. This technique is being used by Facebook, Google, Ebay, Yahoo, Amazon and YouTube.

Instead of sending large images to devices and expecting them to adjust their form factors, the server can serve the mobile pages designed for the requested devices.

Several client approaches and techniques are emerging.

**Client:**

1) **Responsive Design:** This technique involves fluid layout design so the site can adjust according to the resolution of the devices. It mainly adopts a flexible grid, (which implies scaling with the resolution rather than fixed pixel dimensions), CSS and media queries. By using these techniques, it is possible to serve a single webpage to a wide range of devices and have the local client process and render the appropriate designs.

2) **Progressive Enhancements:** The idea of this paradigm is to serve a basic mobile friendly HTML page to all the devices and the JavaScript code will add the functionality progressively to the page.

A client application can take advantage of both these techniques to render webpages on the mobile device.

**Hybrid approach:** You get the best of both approaches. Device detection happens on the server and serves the mobile friendly HTML pages. The experience is then optimized on the client with progressive enhancement and responsive design techniques.

**Application upgrades:** Upgrades of hybrid apps can be tricky since the content can exist on the web and the native parts of the application. Upgrades become mandatory with a change in the native content or in the native wrapper libraries. Changes in the web content may also necessitate an upgrade if the web content is embedded as part of the application.

**Navigational concerns:** This is one of the key considerations for hybrid applications as the UI flow can be complicated. Seamless navigation across the native -> web -> native flow is very much essential. It is important to choose a solution that provides capabilities to address these concerns.

**Data sharing aspects:** As mentioned earlier, along with the navigation, data flow can also happen across components. Application developers should design a channel for sharing the transient data across the native -> web -> native transitions.
**Architectural approaches**

There are different approaches to developing hybrid applications. The overarching theme is to provide access to native device features and web content within the same application.

a) One approach involves using the native web view/control offered by the platform vendors to render the web content and then leveraging the wrapper bridge to access the device features using JavaScript libraries.

Currently, vendors are providing a web view option to render web content, while access to native device features is not directly extended. Several companies found a work around to leverage the native features using JavaScript libraries. In Windows 8.0, Microsoft provides access to the device features without the need for any 3rd party libraries.

b) A custom hybrid web container that leverages the WebKit engine, which can provide access to device features via messaging. It contains a local data store.

In the custom hybrid web container, a dedicated channel is established to access the native features via messaging. This technique gives developers better control as if they were building the app as a native component. The problem with this approach is that the containers must be developed for different platforms and will become platform dependent. They are tied to different WebKit engines (iOS, Android – WebKit, Microsoft – Trident). They also need to be constantly checked against newer platform versions and features.

The following discussion mainly focuses on the first option.
Hybrid apps use a web view control (UIWebView on iOS, WebView on Android and others) to present the HTML and JavaScript files. WebView uses the native browser rendering engine (not the browser itself). The HTML and JavaScript used to construct a hybrid app is rendered/processed by the WebKit rendering engine.

The key ingredient of the hybrid apps is the implementation of an abstraction layer that exposes the device capabilities (native APIs) as JavaScript APIs. This is something not possible with Mobile Web implementations because of the security boundary between the browser and the device APIs.

Several vendors are developing their own JavaScript abstraction layer over native APIs such as TCS (iOS, Android, BB and WP 7.5)

When the native wrapper is compiled around the HTML, CSS and JavaScript resources, there is an interop library added that connects the JavaScript APIs with the platform specific APIs.

### Recommendations

The choice of selection of the hybrid approach should be carefully determined by considering various aspects. As a rule of thumb, if the audience base is wide and the platform diverse, and one needs to deal with device diversity and form factor challenges, a Hybrid approach is highly recommended.

Likewise, if your time to market is low or the total cost of ownership is significant, the Hybrid App approach will help.

If performance is the key concern or UI experience is the paramount factor, the Hybrid approach may not be the right choice. If you have multiple applications on the development horizon with diverse platforms but need a uniform user experience across the web while planning to take advantage of the device features, it is prudent to take a hybrid approach.
For companies who are struggling to find their user base and to make investment decisions, it would be great to start with the simple, cost effective hybrid approach to open the doors for a wide audience base, then decide whether developing platform specific releases would be prudent based on the analytics collected.

As you can see in the above chart, the hybrid approach supports most features.

<table>
<thead>
<tr>
<th>Key Criteria</th>
<th>Web App</th>
<th>Native App</th>
<th>Hybrid App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplatform support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access to hardware sensors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access to peripheral SDKs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native look and feel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>App search through app store distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upgrades</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backward/forward compatibility</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulators</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debugging</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to market</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total cost of ownership</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content Search</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paid apps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advertising</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analytics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique interactive game like experience</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: Full Support | Some Support | Limited/No Support
Conclusion

One should choose a mobile application development approach by taking current and future needs into consideration. It is very difficult to take every possible consideration into account as the mobile industry evolves across platform diversity, disparate devices, network needs, and a more demanding user experience. The mobile market is quickly growing and the only constant is change. Device, platform and browser technology will continue to change each year making it almost impossible to develop a strategy that’s future-proof beyond a few years. The time to market is extremely important and businesses are forced to move forward. Considering all possible variables, it would be prudent to adopt an approach where you invest less in the beginning, yet address a wide variety of audience, devices and platforms and evolve progressively.

The best that developers can do is to ensure that their applications have a good separation of layers. Loose coupling will make it easier to perform the inevitable changes that will be required when mobile technology evolves. With the Hybrid approach, one does not close the door on using the rich, native device features or the native UX experience (wherever applicable). If the user experience is of paramount importance, the native portion can be extended. Another factor to keep in mind is the availability of skilled web developers in the market to bring solutions to the market faster. The hybrid approach provides maximum flexibility, without sacrificing native specific features.

References

3. The (Not So) Future Web, 21 June 2011 Gene Phifer, David Mitchell Smith - Gartner
6. Web site: http://mobilehtm5s.org/
About TCS Mobility

TCS Mobility delivers best in class mobility services and solutions with complete mobility lifecycle consultation and development service along with customization on top of pre-built solutions to provide the best in class benefits to our customers.

Our passion for providing the very best and comprehensive mobility services and solutions to our customers is realized through our deep expertise in mobility gained through the experience of a strong team that has a career long experience in mobile technologies and a dedicated mobility user experience design team that is committed to leveraging the unique native capabilities of each device platform.

Contact

For more information about TCS' Mobility contact mobility.solutions@tcs.com

Subscribe to TCS White Papers

Feedburner: http://feeds2.feedburner.com/tcswhitepapers

About Tata Consultancy Services (TCS)

Tata Consultancy Services is an IT services, consulting and business solutions organization that delivers real results to global business, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled infrastructure, engineering and assurance services. This is delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development. A part of the Tata Group, India’s largest industrial conglomerate, TCS has a global footprint and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at www.tcs.com

IT Services
Business Solutions
Outsourcing