Survey on Cross-Platforms and Languages for Mobile Apps

André Ribeiro
Instituto Superior Técnico
Lisbon, Portugal
andre.ribeiro@ist.utl.pt

Alberto Rodrigues da Silva
Instituto Superior Técnico
Lisbon, Portugal
alberto.silva@acm.org

Abstract — **Nowadays mobile applications are becoming increasingly more present in our daily life, allowing people to perform several tasks through the use of smartphones, tablets or equivalent devices. Despite the great benefits in terms of innovation and in the variety of available solutions, the rapid and continuous growth of the mobile market has resulted in some fragmentation of the platforms that support each mobile device. The existence of different mobile operating systems with different programming languages and development tools can be a problem when someone wants to release an application in as many platforms as possible. The typical approach of simply rewriting the application for each one of those platforms is usually impracticable either in terms of budget or development time, requiring a bigger effort to be made. Therefore, a solution that could generate an application for several platforms (multi or cross-platform) without compromising the overall quality of the application would decrease the time to market of the application and increase enormously the number of potential users. Providentially, during the last years some effort has been conducted to tackle this problem, especially with the emergence of tools and frameworks that facilitate the development of cross-platform mobile applications. This paper focuses on these technologies and attempts to provide a global view of the actual state of this area.**

Keywords: *Application, Cross-platform, Domain Specific Language, Mobile, Model-driven development.*

I. Introduction

Over the last years we have been witnessing a great evolution of the mobile computing industry with the emergence of new devices increasingly more powerful and operating systems with better functionalities [1][2]. Therefore, mobile devices have become more present than ever in our daily life tasks. The common tasks of making calls and sending text messages are being backgrounded by others that make use of GPS, accelerometer, video or audio. The existence of these built-in features along with the applications that use them, make mobile devices, such as smartphones or tablets, very desirable and popular nowadays.

All of these developments resulted from the intense competition among the major companies that dominate the mobile market, namely Apple, Google, Microsoft and RIM. Over the last years these companies developed their own platform with specific tools and application market (see **Table 1**) [3].

Although this competition has enabled the large and rapid growing of mobile market and the emergence of increasingly better features, it was also responsible for a certain fragmentation of the operating systems that support each platform [4].

Table 1 - Major mobile platforms.

Vendor	Operating System	Programming Language	Development Environment	Application Store
Google and Open Handset Alliance	Android	Java	Eclipse	Google Play
Apple	iOS	Objective-C	Xcode	iPhone App Store
Microsoft	Windows Phone	Visual C#/C++	Visual Studio	Windows Phone Marketplace
RIM	Blackberry OS	Java	Eclipse	Blackberry App World

This fragmentation becomes a more serious problem when someone wants to develop an application for multiple operating systems. Due to the specificity of each platform, an application developed for a given operating system is incompatible with the others. This lack of compatibility forces the developers to rewrite the application for each one of the target platforms increasing the effort and the time to market of that application.

Fortunately, some effort has been made over the last years to address this problem, mainly with the emergence of mobile cross-platform tools or frameworks. Essentially, these tools consist in software that allow applications to be created and distributed to multiple platforms, reducing the incremental cost per platform and maximizing the code reuse [1][5]. Also, Model Driven Engineering (MDE), one of the emerging areas of Software Engineering, proved useful in solving this fragmentation, specifically through the possibility of defining apps recurring to domain models and Domain Specific Languages (DSLs).

Thus, this paper aims to describe and analyze some of the existing mobile cross-platform tools highlighting their pros and cons, providing a global view of the actual state of this area.

There were also considered several factors to provide a better distinction between each tool: technology approach, supported platforms, development environment, type of the resulting app and device API support. Specifically, the technology approach has permitted to divide the surveyed tools in four categories: Runtime, Web-to-native wrapper, App Factory and Domain Specific Language.

In this paper, after the introduction, section 2 gives a brief definition of MDE and DSLs. Section 3 describes each one of the surveyed tools. Then, in section 4 is performed an analysis and comparison between those tools. In section 5 are discussed the main strengths and weaknesses of each tool, which is followed by the main conclusions of this survey, in section 6.

II. MDE and DSL

MDE tries to mitigate both software complexity and platform fragmentation problems. In essence, MDE is a software development methodology that seeks to move the source code development process to a more abstract level of specification recurring to the use domain models. These models consist in abstract representations of concepts specific of a certain domain problem. Its main goal is that the model guides all the development activities resulting in quality improvements, increased productivity [6] and shorter time to market [7]. One of the greatest benefits of MDE is the ability to specify the structure and the behavior of a software system in a more platform agnostic way than the traditional programming approaches [8].

A DSL is a language targeted to a particular domain that aims to describe a system using high level specifications. DSLs are supported by a compiler that generates an application from a specification. Since a DSL is expressed using domain concepts, it is normally easier to read, understand, validate and communicate with. In the literature is mentioned that DSLs can improve productivity, reliability, maintainability and portability [9][10][11][12]. DSLs can be either textual or visual. Further on, will be presented two examples of textual DSLs focused on the development of mobile applications.

III. Surveyed Tools

For the survey provided in this paper were considered six tools available online. The main criteria used to choose these tools were the existence of enough documentation and the gratuity of them, excepting DragonRAD. Moreover, the first four tools were selected due to their popularity and extensive use. The remaining two, mobl and mdsl, were selected because they are DSLs and can therefore provide a different approach to solve the fragmentation problem, unlike other similar surveys [5][6][24]. Below it will be described each one the tools.

A. *Rhodes*

Rhodes is a framework for managing enterprise mobile apps and data, developed by RhoMobile. Rhodes allows the development of applications using the well known Model View Controller (MVC) architecture. With this architecture it clearly separates the business logic (controllers) and data definitions (models), written in Ruby, from the visual interface (views), written in HTML, CSS and JavaScript [6]. Rhodes also provides access to native device features through an abstracted set of Ruby APIs.

Rhodes features an app generator that creates controllers and views based on the models defined by the user and that provide basic listing and editing of data. This fact enables that optimistically most of development will be focused on optimizing the views.

When finished, Rhodes apps are compiled into Ruby 1.9 bytecode, which is then interpreted by the Ruby Virtual Machine for the target mobile platform. Therefore, Rhodes is basically a runtime that sits atop the native mobile operating system.

Rhodes also provides an integrated development environment (IDE) based on Eclipse called RhoStudio. RhoStudio facilitates the development of applications with build tools, templates, simulators and debuggers.

Rhodes is part of a wider suite of tools for mobile development (RhoMBUS) that also includes RhoSync for synchronization and back-end integration, RhoHub that provides a cloud-based development environment and deployment via browser and RhoGallery for hosted cloud management of apps [14].

According to reports of RhoMobile there have been over 100.000 downloads of Rhodes and hundreds of Rhodes-based apps on the Apple App Store [5]. Rhodes supports Android, iOS, BlackBerry, Windows Phone and Symbian, and it is available for free under the MIT license [14].

B. *PhoneGap*

PhoneGap is an open source framework developed by Nitobi Software, company acquired by Adobe in October of 2011. PhoneGap allows the creation of mobile apps through the use of HTML5, CSS3 and JavaScript. Therefore, it targets mainly web developers who want to create native smartphone apps and distribute them in the corresponding app stores [15].

Essentially, PhoneGap is a “web code wrapper”, i.e., during the build process the web code is packaged together with the PhoneGap library into a native application execution shell [5]. Therefore, the resultant apps of PhoneGap are hybrid because they are not purely native neither purely web. Not purely native comes from the layout rendering that is done via web browser instead of using the native language and objects of the platform, whereas not purely web-based results from the lack of support of some HTML [4][6].

PhoneGap does not provide a unique IDE to develop applications on all mobile OS, instead it provides a service called PhoneGap Build that allows developers to compile their PhoneGap apps in the cloud. Fundamentally, the developers upload their app written in HTML, CSS and JavaScript and get back app-store ready application binaries. With this service there’s no need to install mobile platform SDKs to build native applications.

PhoneGap is widely used, and by November 2011, Nitobi claimed 600.000 plus downloads of PhoneGap and thousands of apps created with the SDK [5]. PhoneGap supports Android, iOS, BlackBerry, Windows Phone, Symbian and Bada [4][5]. PhoneGap is freely available online under an Apache License 2.0 [15].

C. DragonRAD

DragonRAD is a mobile development tool developed by Seregon. Just like Rhodes, it focuses on mobile enterprise applications. It is comprised of three major components: Designer, Host and Client.

The DragonRAD Designer is a WYSIWYG drag-and-drop tool based in Lua that runs on a Windows desktop environment. It allows the user to develop the application in a more easy way, reducing the effort of coding and simplifying the maintenance of the code. Nevertheless, scripting can be used when more sophistication is required.

DragonRAD applications are compiled into Lua bytecode, which is executed by the interpreter in the DragonRAD runtime environment. Then, the application can be published to DragonRAD Host, a server that fills the gap between enterprise databases and mobile devices, managing the communication and synchronization between them. It supports several database systems such as MySQL, Oracle or SQL Server.

The DragonRAD Client behaves like the native application on device, which contains the code to interpret and run the application created in the Designer. DragonRAD comes with a default client that is loaded in the emulator or the device while debugging. However, a custom client can be build using the ClientBuilder service in DragonRAD's website when the application is finished [16].

Seregon claims 3.000 registered users. DragonRAD supports Android, BlackBerry and Windows Mobile (predecessor of Windows Phone) [5]. Moreover, DragonRAD offers a 30-day trial for the Designer, after that a subscription of $4900 per year is required [16].

D. Appcelerator Titanium

Appcelerator Titanium is a platform for developing mobile, tablet and desktop applications using web technologies developed by Appcelerator. Titanium applications are written in HTML, JavaScript and CSS with development tools and support for PHP, Ruby and Python. It provides JavaScript APIs that use native UI and platform APIs to access UI components such as menus, navigation bars or dialog boxes, and native device features.

Titanium links JavaScript to native libraries, compiles it to bytecode and then the platform SDK (Android or iOS) builds the package for the desired target platform in order to run it either on the native simulator or device for testing, or for final packaging for distribution. The output application contains mostly native code and rendering is executed natively. In addition, the output application contains a JavaScript interpreter runtime and a Webkit rendering engine.

At runtime, the loading performance is lower, as interpreting the source code on the device needs to be done every time the application runs.

Titanium also includes an Eclipse-based IDE called Titanium Studio that offers features like code completion, inline debugging and the possibility to run and deploy applications [1][5][17].

Appcelerator claims a developer client base of 250.000 mobile developers [5]. Titanium supports iOS, Android and Blackberry (beta version) and it is available for free under an Apache License 2.0 [17].

E. mobl

mobl is a free and open source domain specific language developed by Zef Hemel as part of his Ph.D. thesis project. It has a similar syntax to JavaScript and was designed specially to speed up the development of mobile applications. mobl uses HTML5, JavaScript and CSS and offers a language to built native-feeling web apps that can be deployed on several platforms. Its compiler automatically generates the related static JavaScript and HTML files ready to be deployed to any web host and to be cached on the device, in order to enable offline web applications.

mobl allows the definition of user interfaces, data model, application logic and web service interfaces in a declarative manner. It contains language features like entities, screens, controls, styles and web services that makes it domain specific.

It is supported by Eclipse IDE providing features like error detection, syntax highlighting reference resolving or code completion. Also this IDE compiles the code whenever the user saves it allowing it to be readily tested in a mobile browser. mobl supports iOs and Android [18][19].

F. Canappi mdsl

mdsl is a domain specific language developed by Canappi focused on describing an application in a technology and architecture independent way. Similarly to mobl, mdsl allows the definition of a mobile application in a declarative way. The main concepts of this language are entity, view, layout, action and connection.

A view represents the unit of interaction, i.e., the screens. A layout is a simple definition of a group of controls, such as buttons or text boxes, that are connected to views. Some controls may be assigned an action with some custom code or use a standard action, such as "navigate to" another view.

Connections are HTTP based operations which may accept a series of parameters and return a JSON or XML based response. Connection's parameters and response elements can be mapped to the controls using a mapping element. In essence, connections represent web service calls.

Entities represent the data model of the application. The definition of entities generates MySQL schema, standard PHP files and some basic HTML forms which can be used in a companion web site to access the same data as the mobile application.

The development of mdsl apps can be done using Canappi's Eclipse editor. It's also possible to develop using design tools like Apple Interface Builder or Balsamiq and generate an mdsl template from it. A mdsl-based app can be created in five steps: (1) create the msdl file in Eclipse; (2) upload the file in Canappi's website; (3) download the generated code; (4) create the Xcode (for iOS) or Android project; (5) add the required libraries and compile it.

mdsl supports the generation of native iOs and Android (beta version) applications and has plans for a future Windows Phone support. In addition, it is freely available online [20][21].

IV. Analysys and Comparison

This section will emphasize the differences between the tools previously described. The factors considered to analyze and compare the tools were: the technology approach, supported platforms, development environment, type of the resulting app and the device API support.

A. *Technology Approach*

An important factor that distinguishes the tools of each other is the technology approach used (see **Table 2**). In this survey were identified four classes of approaches: Runtime, Web-to-native wrapper, App Factory and Domain Specific Language (DSL).

Runtime consists in an execution environment and a layer that makes the mobile app and the native platform compatibles. Runtimes uses several methods of code execution, such as interpretation, translation or virtualization. The typical approach consists in translating the code to bytecode and then, at runtime, that bytecode is executed by a virtual machine supported by the mobile device. Rhodes and Titanium belong to this category.

Web-to-native wrapper is the approach used by PhoneGap and consists in a solution that packages web code with libraries inside a native app shell. Those libraries have the task of linking the web code to the native one.

App Factories consist in visual tools that allow users to develop their app without code. Usually, they are based in templates and provide drag and drop features to generate the code. This approach is the one used by DragonRAD [1][5].

Finally, DSLs are programming languages or executable specifications focused on a particular problem domain and that use abstractions and domain concepts to describe the apps [9]. mobl and mdsl fall into this class through the application of concepts of the mobile domain.

Some other categories could have been considered in this survey, such as source code translators or cross-compilers that convert source code into another language (bytecode, native language or assembly) and are usually used with Runtimes. In [22] is presented an example of such a tool.

B. *Platform Support*

All of these tools share the benefit of allowing the generation and deployment to multiple platforms from a single codebase. However, as can be observed in **Table 3**, PhoneGap and Rhodes are the tools that support more platforms. In the opposite side, DragonRAD, mobl and mdsl support only two platforms [14][15][16][17][18][20].

Table 2 - Comparison of some development features.

MDE?	Tool	Technology Approach	Language	Resulting App
✗	Rhodes	Runtime	Ruby, HTML, CSS and JavaScript	Native
✗	PhoneGap	Web-to-native wrapper	HTML, CSS and JavaScript	Hybrid
✗	DragonRAD	App Factory	WYSIWYG and Lua	Native
✗	Titanium	Runtime	HTML, CSS and JavaScript	Native
✓	mobl	DSL	mobl	Web
✓	mdsl	DSL	mdsl	Native

Table 3 - Platforms supported by each tool.

Tool / Platform	Android	iOS	BlackBerry	Windows Phone
Rhodes	✓	✓	✓	✓
PhoneGap	✓	✓	✓	✓
DragonRAD	✓		✓	
Titanium	✓	✓	✓	
mobl	✓	✓		
mdsl	✓	✓		

C. *Development Environment*

In terms of the development environment, all the tools, with the exception of PhoneGap, use a specific IDE to develop the applications, as can be seen in **Table 4**. Four of the analyzed tools use Eclipse or an Eclipse-based solution as IDE. PhoneGap instead offers an extension that could be installed on the native IDE of the mobile platforms (e.g. Xcode for iOS or Eclipse for Android). It also offers the PhoneGap Build to compile the apps.

A different approach is used by DragonRAD that provides a visual drag and drop editor. Like DragonRAD, mdsl also supports the use of visual editors, namely Interface Builder (available only for Mac OS) and Balsamiq [14][15][16][17][18][20].

Table 4 - Development environment of each tool.

Tool	Development Environment
Rhodes	RhoStudio (Eclipse-based) / RhoHub
PhoneGap	Native IDE of the mobile platform
DragonRAD	DragonRAD Designer
Titanium	Titanium Studio (Eclipse-based)
mobl	Eclipse
mdsl	Eclipse / Interface Builder / Balsamiq

D. *Type of the Resulting App*

A great difference between the tools is the type of the app produced, which can be native, web or hybrid. A native app consists in the result of building an app in a device's native programming language. A web app is a website accessed from the device's browser and that is implemented to resemble an application rather than a traditional website. A hybrid app is built using web technologies and then wrapped in a platform-specific shell that allows it to be installed just like a native app [22].

In **Table 2** can be observed that Rhodes, Titanium, DragonRAD and mdsl generate native apps. PhoneGap generates hybrid apps while mobl generates essentially web apps [5][14][15][16][17][18][20].

E. Device API Support

Another important factor to take into account is the ability to access device capabilities such as camera, accelerometer or GPS. **Table 5** summarizes some of the capabilities offered by each tool.

Table 5 - Ability to access device capabilities comparison.

Feature / **Tool**	Accelerometer	Camera	Contacts	Media	GPS
Rhodes	✓	✓	✓	✓	✓
PhoneGap	✓	✓	✓	✓	✓
DragonRAD	✓	✓		✓	✓
Titanium	✓	✓	✓	✓	✓
mobl		✓		✓	✓
mdsl				✓	✓

V. Discussion

The factors or characteristics pointed in the previous section could be crucial to the selection of one tool instead of other. From these factors, it is possible to suggest the more suited target audience of each surveyed tool and to emphasize the main strengths and weaknesses of each one.

By the fact of being Runtimes, Rhodes and Titanium target software developers who seek for more compatibility and proximity to the native look and feel of each platform. Although, this approach could represent a degradation of performance since it's needed to interpret the code at runtime.

In turn, PhoneGap as a Web-to-native wrapper is more suited to web developers who want to convert their web apps to native in order to publish them in the mobile app markets.

DragonRAD, because of being an App Factory, is more suited for non-developers or people that want to develop simple apps without the burden of programming.

mobl and mdsl, as DSLs, are more appropriate for developers that want a language closer or more focused in the mobile domain.

Furthermore, it is possible to differentiate the tools according to the target app. mobl, mdsl and PhoneGap are more targeted to Rich Internet Applications. In turn, Rhodes, DragonRAD and Titanium are more oriented to data driven enterprise apps for B2C, B2B and B2E with backend support and where the need for a more proximity with the native features is quite important.

Following, it will be presented the main strengths and weaknesses of each tool.

A. Rhodes

Rhodes is available for free and supports a broad range of mobile platforms and device hardware features. Also the existence of an Eclipse-based IDE and the support of browser-based app construction using RhoHub represent advantages of using Rhodes. The use of Ruby code helps to structure the business logic using the MVC architecture.

On the other hand, Rhodes requires the knowledge of Ruby, which is a language not as popular as others, such as Java or PHP. Another weakness is the fact that Rhodes doesn't generate native source code, only native package, which can limit any further tweaking of the app. Moreover, the update of HTML or JavaScript code used by Rhodes requires a complete rebuild [1][5][6][14][24][25].

B. PhoneGap

PhoneGap is free and has the widest range of platforms supported from all the surveyed tools. PhoneGap apps are built purely in HTML, JavaScript and CSS what lowers the barrier of adoption for web developers. The providing of all native wrapper source code allows its customization if needed. In addition, PhoneGap interfaces with several device hardware features.

On the other hand, PhoneGap assumes that normal capabilities of a web-based application are available. Moreover, it lacks of support for some native UI components and doesn't provide its own IDE, requiring the use of the IDEs of each platform [1][5][6][15][24][25].

C. DragonRAD

DragonRAD has as main strengths the generation of native apps and the drag and drop editor that lowers the entry barrier for non-programmers.

As main weaknesses, DragonRAD has the license price and the small set of platforms supported. The drag-and-drop development approach can cause limited appeal to IT departments, where control of code or design flexibility is quite important [5][6][16].

D. Appcelerator Titanium

Titanium has as main strengths the fact that it is freely available, the generation of truly native code and the existence of its own IDE. Furthermore, as apps are built purely in HTML, JavaScript and CSS makes lower the barrier of adoption for web developers.

The main disadvantages of Titanium are the potentially restrictive APIs and the small set of platforms currently supported [1][5][17][24][25].

E. mobl

The main strengths of mobl are the free license, its syntax and semantics similar to JavaScript, the use of domain concepts that decreases the code size and the integration with Eclipse.

In the opposite side, its main weaknesses are the fact that users must learn a new language, the small set of platforms supported, the limited documentation and APIs provided and the generation of apps that rely on device browser capabilities to render the interface [18][19].

F. *Canappi mdsl*

The main strengths of mdsl are the free license, the use of domain concepts that decreases the code size, the generation of native apps and the support of visual editors.

On the other hand, mdsl is a new language that users have to learn, it supports a small set of platforms and has limited documentation and APIs [20][21].

VI. Conclusion

This paper presented an analysis between six cross-platform tools to develop mobile apps. Tools like the ones surveyed are becoming increasingly more popular due to their ability to create apps for the major mobile platforms from the same code base. Thus, this kind of tools is revolutionizing the way people develop mobile apps reducing the effort required to produce and publish them on several mobile markets.

The major characteristics of each tool, such as the technology approach, the number of supported platforms or the development environment, were taken into account, not only to detail the tool, but also to compare it with the others, clarifying the reader. From that analysis was possible to point the main strengths and weaknesses of each tool and suggest the more appropriate target audience and use cases of that tool. Another important fact explored in this paper was the use of MDE in the surveyed tools. The great majority of the tools available online do not apply MDE concepts, but two of the surveyed tools use it.

Concluding, lots of cross-platform tools are available online nowadays being the major challenge understand which one is the best to achieve the goals of a certain user or company. Moreover, cross-platform tools are still evolving and just like other software tools have flaws and limitations, but represent a straightforward solution to solve the platform fragmentation problem. A more often application of the MDE concepts in this area may represent a good solution not only to develop cross-platform apps, but also to ease the development and captivate a larger number of users through the use of domain concepts. As future work, an extension to this survey can be made, namely increasing the set of analyzed tools and comparison factors.

References

[1] G. Hartmann, G. Stead and A. DeGani, "Cross-platform mobile development", Mobile Learning Environment, Cambridge, March 2011.

[2] S. Devitt, M. Meeker and L. Wu, "Internet Trends", Morgan Stanley Research, April, 2010.

[3] A. Charland and B. LeRoux, "Mobile Application Development: Web vs. Native", Commun. ACM, vol. 54, no. 5, pp. 49-53, New York, May 2011.

[4] S. Allen, V. Graupera, V. and L. Lundrigan, "Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution", 1st Edition, Appress, New York, April 2010.

[5] Vision Mobile, "Cross-Platform Developer Tools 2012: Bridging the worlds of mobile apps and the web ", February 2012.

[6] M. Book, S. Beydeda and V. Gruhn, "Model-driven Software Development", Springer, 2005.

[7] S. Sendall and W. Kozaczynski, "Model transformation: the heart and soul of model-driven software development", Software IEEE, vol.20, no.5, pp. 42- 45, September-October 2003.

[8] T. Kuhn, R. Gotzhein and C. Webel, "Model-Driven development with SDL - process, tools, and experiences", In Proceedings of the 9th international conference on Model Driven Engineering Languages and Systems (MoDELS'06), pp. 83-97, Springer-Verlag, Berlin, Heidelberg, 2006.

[9] A. van Deursen, P.Klint and J. Visser, "Domain-specific languages: an annotated bibliography", SIGPLAN Notice 35 Issue 6, June 2000.

[10] A. van Deursen and P. Klint, "Little languages: little maintenance", Journal of Software Maintenance, vol. 10, no. 2, pp. 75-92, March-April 1998.

[11] R. B. Kieburtz, L. McKinney, J. M. Bell et al., "A software engineering experiment in software component generation", In Proceedings of the 18th international conference on Software engineering (ICSE '96), IEEE Computer Society, pp. 542-552, Washington, 1996.

[12] R. M. Herndon, Jr. and V. A. Berzins, "The Realizable Benefits of a Language Prototyping Language", IEEE Trans. Softw. Eng., vol. 14, no. 6, pp. 803-809, June 1988.

[13] I. Singh and M. Palmieri, "Comparison of cross-platform mobile development tools", IDT: Malardalen University, 2011.

[14] Rhodes: Developer Reference. Accessed on April 2012. http://docs.rhomobile.com/rhodes/introduction

[15] PhoneGap: How PhoneGap Works. Accessed on April 2012. http://phonegap.com/about

[16] DragonRAD Help Introduction. Accessed on April 2012. http://dragonrad.com/foswiki/bin/view/Help/Introduction

[17] Appcelerator Titanium Platform Overview. Accessed on April 2012. http://www.appcelerator.com/platform

[18] mobl website. Accessed on April 2012. http://www.mobl-lang.org

[19] Z. Hemel, "mobl: a DSL for Mobile Web Development", InfoQ, March 2011. Accessed on April 2012. http://www.infoq.com/articles/Mobl

[20] Creating a mobile application with mdsl. Accessed on April 2012. http://www.canappi.com/pages/mdsl

[21] J. Dubray, "Challenges and Opportunies in Mobile Application Development And Mobile DSLs", InfoQ, May 2011. Accessed on April 2012. http://www.infoq.com/articles/mobile-dsl

[22] A. Puder, "Cross-compiling Android applications to the iPhone", In Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java (PPPJ '10), ACM, pp. 69-77, New York, 2010.

[23] A. Jones, "Native, Hybrid or Web Apps?", January 2012. Accessed on April 2012. http://buildmobile.com/native-hybrid-or-web-apps/

[24] T. Paananen, "Smartphone Cross-Platform Frameworks", Bachelor's Thesis, Jamk University of Applied Sciences, April 2011.

[25] J. Rowberg, "Comparison: App Inventor, DroidDraw, Rhomobile, PhoneGap, Appcelerator, WebView, and AML. Application Markup Language". Accessed on April 2012. http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml/