Mobile Apps Development: A Framework for Technology Decision Making

Conference Paper · October 2012

DOI: 10.1007/978-3-642-36632-1_4

5 authors, including:

Emiliano Masi
Robert Bosch Start-up GmbH, Ludwigsburg, Germany
1 PUBLICATION **14** CITATIONS
SEE PROFILE

Giovanni Cantone
University of Rome Tor Vergata
93 PUBLICATIONS **961** CITATIONS
SEE PROFILE

Giuseppe Calavaro
IBM
10 PUBLICATIONS **90** CITATIONS
SEE PROFILE

Mobile Apps Development: A Framework for Technology Decision Making

Emiliano Masi[1], Giovanni Cantone[2],
Manuel Mastrofini[2], Giuseppe Calavaro[3], and Paolo Subiaco[3]

[1] FIZ Karlsruhe, Hermann-von-Helmholtz-Platz,
76344 Eggenstein-Leopoldshafen, Germany
emiliano.masi@fiz-karlsruhe.de
[2] University of Roma Tor Vergata, DICII
Via del Politecnico, 1 – 00133 Roma, Italy
{cantone,manuel.mastrofini}@uniroma2.it
[3] IBM Italia SpA
Via Sciangai, 53 – 00144 Roma, Italy
{gcalavaro,paolo_subiaco}@it.ibm.com

Abstract. Developers of a new Mobile App have to undertake a number of decisions, including the target platform and the development technology to utilize. Even though there is no one-size-fits-all solution, which could meet all needs for all contexts, this paper is concerned with an exploratory study aimed to provide developers with a framework to support their technology selection process, including practical guidelines on how to select the technology that best fits the given context and requirements. The exploited research methods are survey, interview, and case study. Results consist in a model of, and a collection of data and experts' experiences about, some advanced platforms. Results are packed in a tool-prototype: once entered the needs and required device features, the tool returns measures that allow a decision maker to identify the development technology, among the recommended alternatives, which best fulfills the actual requirements.

Keywords: Mobile App development, mobile platforms, mobile development technology, mobile development approaches.

1 Introduction

The basic question that this paper tries to answer is: What is the right technology to use for the development of a mobile application for given requirements and context?

In order to develop a new Mobile App, in fact, several decisions have to be made. The first one is the platform on which the application will run. Such decision is usually made by the marketing management. Very often, more than one platform is chosen, although versions for the different platforms can be released at different times. Instances of platforms include, but are not limited to:

- iOS platforms,
- Android platforms.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 64–79, 2013.

As soon as the platforms decision is undertaken, one more very important choice is required, which matches the aforementioned question: What technologies should be used to develop the given mobile app? Now, the Technical Leader has the ownership of such decision. As we will be explaining in the next sections, the most known examples of such technologies include:

- HTML5;
- CSS;
- JavaScript;
- Platform Specific Software Development Kit (SDK), including:
  a) Android SDK, and
  b) Apple® XCode;
- Titanium Mobile, and related supports for Mobile App development;
- Adobe®, and related supports for Mobile App development;
- PhoneGap;
- jQueryMobile;
- Sencha Touch;
- DojoMobile.

Indeed, even when a single platform is initially targeted to develop an app, few ways exist to enact the development, which can be classified in three Development Approaches (DAs):

- Native DA: i.e., pure native development, using the platform specific SDK;
- Web DA: i.e., pure web app (HTML 5 allows Apps that are almost as powerful as native apps);
- Hybrid DA: i.e., using a mixed approach, as supported by several different technologies discussed in the remaining of this paper.

A decision concerning the DA to use has to take into account both functional and non-functional app requirements, such as the need for different devices on board (GPS, Camera, etc), as well as constraints placed by the selected target platforms, schedule, and budget. Therefore, a multi-criteria decision process is required [1][2], and both tools and frameworks would desirable to support and guide it.

2 Purposes of the Study and Research Methods

Evidence-based research in the field of software development for mobile apps is in its beginning and still needs enhancement. Data and knowledge bases are not yet publicly available. Concerning private companies, it may be that data collections are not yet capitalized organization-wide. As a consequence, it is still frequent for researchers to start studies by producing their own data from scratch.

The research methods that we used for this study are survey and case study research. For data collection, we used literature and technical documentation survey, interviews to practitioners, and development of case studies by using and comparing different technologies.

The purpose of this work is exploratory rather than confirmatory [3]; in fact, we do not aim to give a final answer to the basic research question, but we want to check whether an approach is feasible and capable to support answering the basic question. Consequently, in the remaining, no formal goals [4], hypotheses [5] or details concerning the utilized research objects and processes (case studies, handbooks, papers, etc.) will be presented.

In general, we do not aim to address the false problem of defining the "absolute best" technology for mobile software development. Vice versa, we remark that our expectation is that no single development approach will fit all needs of all customers. As Fling observed [6], whatever medium we use to address our goals, we have a number of choices, each with its own pros and cons. Similarly to all other branches of Software Engineering, we expect some mobile media to be quick to create apps, but accessible to a few of the customers and/or delivering hard to maintain apps; others could address a larger market, but far more expensive and complex to use.

However, it is clear that companies are facing an obvious trade-off between user experience and application functionality on the one hand, and development costs and time to market on the other hand. Therefore, the challenge is to choose a development approach and a technology that are capable to balance requirements with the available budget and time-to-market [7]. At this point, the fundamental question is: "What is the right choice?".

In order to help answer such question, this paper aims to provide

(i) a guide to technology decision, and
(ii) a framework to support such decision, which could evolve based on new technologies that will come out.

3 Characterizing the Technologies to Study

The choice of the platform dictates the range of technologies to use for developing an app. If we need an application expected to be multiplatform, then we have a range of technologies and possible approaches to use - besides developing the same app for each platform using their native SDK, - i.e. hybrid or web approaches. Conversely, if we need an app for a single Operating System (OS), then we have different choices available and the right approach could be the native development.

In the remaining, we will try to answer the initial question about mobile technologies by focusing on three Platform Categories (PCs):

PC1. iOS platforms;
PC2. Android platforms,
PC3. Other platforms, which we merge under this category, namely "Others".

As already mentioned, after the selection of the target platform(s), the next step consists in choosing both the Development Approach, and the Development Technology (DT).

There are three DAs available for apps development:

DA1. Web apps
DA2. Native apps
DA3. Hybrid apps.

Concerning the DT, it is not independent from the selected DA and PC. In this paper, we take into consideration a limited set of development technologies, among the ones available on the market. Specifically, we consider those technologies that the largest software companies commonly adopt worldwide. We have already listed such technologies in the previous Section 1. In the following, we consider these technologies and group them by the selected DA.

Web apps: choosing the technology is quite straightforward:

DT1. HTML5, CSS, and JavaScript. Since these technologies fit together, we consider this set as a single choice.

Native apps: the platform specific development kit can be selected, e.g.:

DT2. Android SDK;
DT3. Apple XCode.

Additionally, the following technologies can support the development of Native Mobile Apps:

DT4. Appcelerator®'s Titanium Mobile,
DT5. Adobe® Flex, in conjunction with Adobe® Air.

Last two development technologies can create cross-platform apps from the same source code.

Hybrid apps: PhoneGap can be used, in addition to the following JavaScript frameworks:

DT6. (PhoneGap) + jQuery Mobile;
DT7. (PhoneGap) + Sencha Touch;
DT8. (PhoneGap) + Dojo Mobile.

A brief review of the listed technologies, including their pros and cons, is provided in the remaining of this Section.

3.1 HTML5 + CSS3 + JavaScript (DT1)

Since they are web technologies, the learning curve is expected to be very steep (i.e. we expect developers to quickly get familiar with them), if compared to the native languages. Additionally, when writing an application, multiple platforms can be targeted without encountering any significant problems. On the other hand, since HTML5 and JavaScript are not allowed to access all device features, the developer could encounter some limitations. Moreover, as an application built with such stack has to be interpreted by the browser, performance can be worse than an application developed with native languages.

3.2 Platform Specific Software Development Kit (DT2,3)

It ensures the full exploitation of all features of the platform on which the practitioner decides to develop the app, in addition to the native look & feel. The most obvious downside is the ability to address only one platform; for this reason, if the organization is willing to produce a multiplatform application, the code has to be rewritten for each

chosen platform. This contributes to get high development cost and long developing time. It's important to underline that in the following DT2 and DT3 will be considered together.

3.3 Appcelerator® Titanium Mobile (1.8.1) (DT4)

Titanium Mobile is a cross-platform mobile development framework that enables developers to write JavaScript code, which is then compiled down to native code for the iOS and Android platforms. If the organization wants to build a multiplatform application, Titanium Mobile can save time with respect to developing via platform specific SDK. Additionally, a real native application is obtained, with the platform-specific look & feel.

3.4 Adobe® Flex Mobile (4.5) + Adobe Air (DT5)

Flex is another cross-platform mobile SDK. It enables developers to write their applications by using the ActionScript language for Android, iOS, and Blackberry platforms. Concerning iOS, it generates a pure native application written in Objective C. Concerning Android and Blackberry, the developer has to install a further layer on which the application will run: the Adobe Air Environment. Applications developed by using Flex are deployable on the most important app stores.

3.5 PhoneGap (1.7.0) (DT6, 7, 8)

This technology enables developers to create a web application wrapped into a native container. This means that developers can write an application once, and, when they need to migrate to another platform, they can reuse the code by wrapping the initial web app into another native ad-hoc wrapper. Since both wrappers offer the same interface to developers, no changes are required to the previously written code. Developers still have to face the cons associated to using HTML5, CSS, and JavaScript (see Sub-section A above) but, at the same time, they receive some of their benefits, including a steep learning curve, the possibility of deploying the app on appstores, and the chance of extending the native wrapper to use device features not available in the default HTML5.

Since PhoneGap allows to develop pure web apps, selecting a JavaScript framework appears to be an essential choice. In this paper, we take into account three main JavaScript frameworks, among the ones available on the market, i.e.: jQuery Mobile, Sencha Touch, and Dojo Mobile. jQuery Mobile seems to us to be a good candidate for very basic and logically simple applications, while Sencha Touch, due to the different development approach, has to be preferred when the organization is willing to leverage more complex development patterns, such as the Model-View-Control. It seems to us that this 1^{st} technology shows a very flat learning curve, but it gives practitioners the possibility to keep under management the development of even the more complex applications. Dojo Mobile seems to us to be the most complete technology, among the three considered above, as it provides both development approaches offered by jQuery and Sencha. However, if a simple business logic, small footprint, and very small time to market are needed, jQuery Mobile could be the right choice.

4 Drivers for Technology Selection

In our study, the technology selection is driven by a main set of requirements, grouped by the following two categories:

(i) Needs
(ii) Device features.

Based on literature [6], [7-16], we can breakdown these categories as in the followings.

Needs represent both functional and non-functional requirements. Needs, or, equivalently, Non-Functional Drivers (NFDs) are:

NFD1. Access to native hardware features;
NFD2. High performance;
NFD3. Cross platform;
NFD4. Easily upgradeable;
NFD5. Deployable on app stores;
NFD6. Small footprint;
NFD7. Quick coding and prototyping;
NFD8. Complex
NFD9. Simple business logic;
NFD10. Custom look & feel;
NFD11. Platform look & feel;
NFD12. Cheap Development cost.

Device Features (DFs), as commonly found on mobile devices, include:

DF1. Accelerometer;
DF2. Compass;
DF3. Orientation;
DF4. Light;
DF5. Contacts;
DF6. File;
DF7. Geolocation;
DF8. Media;
DF9. Network;
DF10. Notification Alert;
DF11. Storage;
DF12. Multitouch;
DF13. SMS;
DF14. Bluetooth;
DF15. Video Capture.

5 A Model for Evaluating Mobile Technology

Based on the content of sections 3 and 4, answering to our basic question − What is the right technology to use for the development of a mobile application for given requirements and context? − means to solve a problem in a space made by forty dimensions: 3 related to PC, 3 to DA, 7 to DT, 12 to NFD, and 15 to DF. Three of

these dimensions, the PC-related ones, are alternative dimensions; the same holds for the three DA-related dimensions (this could be used to merge them in two dimensions and downsize the space to 36 dimensions). The twelve NFD-related dimensions are independent one another, and the same holds for the fifteen DF-related dimensions; however, DF dimensions can depend on NFD, DA, and PC dimensions, even though we might still be unable to express, a priori and formally, such dependencies. Additionally, an application is required to meet many needs, a device feature can serve many needs, and each development technology can offer different features, depending on the platform it is used for. In other words, we are not coping with a system that we can immediately and easily model via mathematical functions, assuming it is feasible.

To manage the complexity, our decision was to proceed by abstraction and classification, and we eventually consolidated those dimensions in five macro-dimensions. This led to constructing a five-macro-dimension space of: Needs, Device Features, Development Technologies, Platforms, and Development Approaches. The domain of each macro-dimension includes the items listed in the previous sections for PC, DA, DT, NFR, and DF. Since we see no interest in sorting those items, in their domains, they are listed in arbitrary order, so obtaining a Nominal scale [5], [17], [18] for each dimension.

This discrete space of five macro-dimensions can be easily and automatically generated and updated whenever new platforms or development technologies/ approaches appear on the market or are removed, for any reasons, from the set of accepted candidate technologies/approaches.

Our next step is to represent: (i) each technological stack as a point in this space, according to its capability to meet some requirements (e.g. support for a given device); and (ii) the app to develop as a point in the same space, according to the importance that each requirement has with respect to the app context (e.g. importance of the presence of a given device).

This way, a proximity measure could be created to quantify the distance between the point representing the app and each point representing a technological stack. The technological stack which minimizes the distance from the app is the first candidate to be the best fit to the considered app context. The feasibility of mapping technological stacks and app requirements as points in the previously defined space can follow the impact vector model [19]; in particular, the app requirements can be modeled as a "goal impact vector" (i.e. the expected result of the development), while technological stacks can be modeled as "strategy impact vectors" (i.e. the possible combinations of languages, approaches, platforms and tools that can be used for the development) [19]. For brevity, we omit the details on how to map the app and the technological stacks to points in the space. However, a proximity measure still has to be defined, so that a Tech Expert can apply our model in a real context; a definition of such proximity measure and an example of use of our model are provided in section 8.

6 Implementation and Graphical View of the Model

There are both numerical and graphical problems to represent a space with more than three dimensions. Concerning the first, discrete dimensions and Ordinal scales rather than Real scales are possible for the interesting functions, with consequent limits on

the mathematics that can be applied when looking for optimal solutions under defined constraints. Concerning the last, our decision was to start by using a simple user interface; in fact, the focus of this paper is on (i) exploring the mobile development technology modeling for decision-making; and (ii) empirically verifying models in lab and on the field. This is the reason why we did not put much effort on building up an advanced user interface for our tool prototype. In particular, the prototype is based on a table (see Table 1) which implements the space defined in section 5. In order to model the five dimensions in a flat surface – which allows an easy use of the model, – the item type of such table (i.e. a point in the space) represents more than one piece of information; specifically, each item of the table is itself multi-dimensional, so creating a topological space similar to a manifold [20]. Each elementary item of the table (i.e. each dimension of a given point) expresses a measure in a four-value Ordinal scale, which are graphically represented by the symbols  ,  ,  , and  , where denotes the null value, i.e., “Not supported”.

In practice, the table represents a complex but homogeneous abstract entity, i.e. a 5-dimensional hypercube, whose elements are symbols of the given ordinal scale.

Table 1 is structured in two quadrants. These realize two guides for selecting the technology to use according to: a) the needs, and b) the device-specific hardware features that the particular technology offers, respectively.

The first quadrant implements the subspace (DT, NFD, DA, PC); the abscissa reports the development technologies, the ordinate represents the needs. The type of development approach, as leveraged by the corresponding development technology, is represented by different columns and different background colors: (i) Web in the first column (Orange); Native in the subsequent triple of columns (Blue); and Hybrid in the last triple of columns (Green). Each item shows a value in the given Ordinal scale. Since the values reported do not depend on the platforms specificities for each given development technology, such platforms are not represented explicitly. In practice, this quadrant represents a cube with colored slices; each element of the cube includes as many times the ordinal measure, shown in the first quadrant, as the number of platforms.

The last quadrant shares the abscissa (DT) with the first quadrant, while the ordinate represents the device features. Many items in this quadrant represent arrays of platforms (rather than any platform as in the first quadrant). The reason is that each technology can offer different features depending on the platform on which it is used, and a measure is expected for each of those platforms. Again, the actual platforms that are considered separately are iOS and Android, whereas the remaining platforms are grouped into the category “Others”.

The scale elements assume a slightly different meaning in the two quadrants. In the first quadrant, the semantic of the scale is {Null, Insufficient, Sufficient, Excellent}; in fact, the symbols indicate the extent to which, for the considered development approach, that specific technology satisfies the corresponding need:  insufficiently,  sufficiently, and  excellently. In the last quadrant, instead, the semantic of the scale is {Null, Rough, Medium, Well}; in fact, those symbols indicate whether the specific technology provides APIs for using the corresponding device feature, and their level of support: “Well”, i.e., supported for all the different versions of the same platform, “Medium”, i.e., supported for some but not all the different versions of the same platform”, or “Rough”, i.e. lightly supported. In the first quadrant we can find

only one symbol per table item, whereas in the last quadrant, as already mentioned, an item is an array of symbols (i.e. a 4-dimension point), and can also include all symbols together (one per point dimension).

We filled out the dimensions of the matrix discussed above by collecting information from various scientific papers, personal experiences (first quadrant), and technical online documentation (last quadrant). Table 1 also synthesizes on the results from this work.

7 The Process of Technology Decision Making for Mobile

As already mentioned, we want to develop a guide to select the best technology to use for the development of a specified mobile application in a given context. For this reason we can say that our output is the Technology dimension, while the other dimensions of Table 1 are our inputs.

Probably, a multivariate analysis should be used for identifying the best technology to exploit under the given requirements and constraints. However, at this stage of our work, since there is no way to apply multivariate with the small and artificial dataset available, our decision was to start by using simpler techniques.

In general we have two kinds of requirements: General Requirements (GRs), which are present in any mobile app, and Specific Requirements (SRs), which are explicitly stated for the current app, and are subtypes of the GRs.

In order to enact a technology selection process, the Tech Experts are required to assign a weight to each GR, based on its importance or relevance for any app to develop, and then to give a score to each requirement. Concerning the latter, let N be the number of points to distribute across needs and device features. In the setting that we used for our approach, 25% of such points are automatically and evenly assigned to the GRs, and the remaining 75% of the N points are left for assignment to Tech Leaders; these will assign them as they prefer to the SRs, based on their assessment of the requirement relevance for the app being developed. The initial 25% assignment is made to diversify the obtained results, without losing the contributions that other features give to the decision making. This point distribution can be changed arbitrarily, without affecting the model validity.

Finally, Tech Leaders enter Table 1 and obtain, as a result, a value in the previously defined Ordinal scale for each technology. In order to get more manageable results, they can make the further decisions of translating those values in the notation of Real numbers, e.g., by using a locally defined translation map; this allows to switch from an Ordinal scale to a Real measurement model, which enables to apply the algebra of Real numbers. At this point, each generated number represents a numerical score which should quantify numerically the capability of each technology to fulfill any given requirement.

We are aware that the mentioned scale transformation is a theoretical and practical hazard [17,18]. However, also in this case, the impact vector model supports us to combine and manage heterogeneous dimensions [19]. Since the goal of this paper, as already stated, is not to provide a formal and final model for technology selection, the mathematical foundation of our model, including the scale transformation rationale, is

not herein detailed. Nevertheless, practical evidence of the model validation and a case study are reported in the remaining sections, while the definition of a formal model is deferred to a shortly coming future work.

8 Case Study

As an example of using Table 1 for decision making, let us consider a synthetic version of a case study we conducted.

Let us suppose that a Tech Leader is requested to develop an application with the requirements shown in the Table 2, row 1.

Ten features are explicitly stated as significant for this application, as shown and numbered from 1 to 10 in Table 2, row 2. Let the Tech Leader assign the same weight to all general requirements (GRs), and assuming N=1000, to distribute the remaining 750 points on each specific feature (SRs) in the following way:

1. 100 points, 2. 30 points, 3. 10 points, 4. 60 points, 5. 150 points,
6. 30 points, 7. 80 points, 8. 70 points, 9. 70 points, 10. 150 points.

Additionally, in order to have manageable results, let the Tech Leader map the given Ordinal scale into a Real scale, as explained in the previous section, by using the following map: Excellent: 2.0 points. Sufficient: 1.0 point. Insufficient: 0.5 points. Not Supported: -1.0 point. This gives a Real scale in the range from -500.0 (worst case, i.e. Not supported in every dimension) up to 2000.0 (ideal technology for the given requirements, i.e. excellent in every dimension.)

The Tech Leader can now enter Table 1, and obtain an ordinal score for each technology. Subsequently, he/she can enter the map and translate the ordinal symbols found into Real numbers; eventually the results shown in Table 3 will be obtained.

As we can see, the two technologies with the highest scores are:

1. Platform Specific Development Kit with 1675.9 points.
2. Adobe Flex Mobile + Air with 1491.05 points.

For the given context, the more meaningful definition of proximity measure is the vector distance from the optimal vector (i.e. the resulting difference vector). The optimal point is the one with maximum achievable score, i.e. 2000 points, which exactly matches the required characteristics of our app. In practice, for our context, minimizing such distance is equivalent to pick the maximum among the computed technology scores. Based on the scores shown in the Table 3, the best technological choice for this application should be the native approach with the Platform Specific SDK. A good choice would be also the Adobe Flex Mobile + Air, in fact, if there are heavy constraints on the time to market and there is not enough in-house skills for each platform (which is a characteristic not represented in our model), such technology could be the only feasible way.

In both cases, however, developers are given all the information required to make an informed choice on what technology best fits the application to develop in the given context.

9 Validity Issues, Solution Limitations, and Lesson Learned

The model proposed in the previous sections was populated by interviews with experts of, and tested in four case studies within, the IBM Italy Rome Smart Solutions Lab. Such case studies were concerned with different problems to address and they covered a wide range of possibilities, but in a limited number of domains (e.g., mobile banking, education, and mobile marketing).

Once minded that we are coping with an exploratory study, based on the feedback that we had from practitioners, the solution validity of the study, i.e., its *internal validity*, can be considered high enough; on one side, this is concerned with the solution *usefulness* (does it address relevant practical problems?), which was very positively evaluated by the involved experts; on the other hand, it is concerned with the *goodness* (how well the solution is constructed); in the opinion of the same practitioners, practical and actionable results were produced by the study, even in absence of a strong and theoretically rigorous framework.

In our view, the proposed solution is *portable* to, and promises to be *effective* in, different contexts and cases. In other words, the *external validity* of the proposed solution is considerably high. However, due to the exploratory nature of the study, consequential *limitations* should be taken into consideration.

First of all, it is important to remark that interviews, as source of collected data and knowledge, involved quite a limited number of participants in the role of mobile technology experts. Also, only one subject developed the four case studies that we have been performing. Additionally, the proposed model was piloted with a single organization. This increased the threat of having the same person repeating all positive or negative actions from case to case, and people doing the technology evaluation in the same context.

One more aspect, which was not considered in this paper, and which affects the external validity and limits of the proposed solution, relates to the communication model utilized by the app to develop, e.g., apps connecting more than one partners synchronously and/or asynchronously. This includes considering if, and to which extent, media-content/social networks would influence the development of mobile apps, and how this should be modeled by an enhanced version of the proposed solution.

Again concerning threats on validity and limits of the proposed solution, we notice that our model is oriented to supporting project leaders in decision making, but it is based only on technical attributes. Constraints placed by the strategic management should be introduced in the model, to support technicians to undertake value-based decisions [21], including risk and return on investment, and proceed coherently with the strategic goals of their company.

A further relevant aspect is that the information gathered during interviews was mainly qualitative knowledge; additionally, measures in Ordinal scale, as produced from case studies, were eventually influenced by the subjectivity of the involved experts.

Also, one more threat to internal validity is the lack of a complete formal model supporting the proposed solution. Indeed, the discrete macro-space, which this paper has provided to support decision-making for mobile development, is not a Euclidean space, so, in general, we cannot add scores up as easily as we did in the case study described in section 8. Also, we have not considered interdependent choices of technologies, mutual influences etc.

In our understanding, in case of exploratory studies, which is our case, it is reasonable to accept all the aforementioned validity threats, given the mitigation strategies we enacted, as usual in the experimentation in the field of Software Engineering [5].

For what concerns the lesson learned, let us recall that interviews with experts helped populate Table 1. However, as the case studies proceeded, we had to include additional attributes in the model (i.e. dimensions in the space) and refine the seeded values. This confirmed our conviction that an iterative-incremental approach should be enacted to define the model and populate its base of knowledge. Following the initial training of the data and knowledge base, as enacted by technology experts, the enrichment of the base should be enacted continually, based on decisions made by the organization's Tech Leaders, possibly leveraging a structured methodology, e.g. the Quality Improvement Paradigm [22]. Additionally, each development organization might need to manage its own base of knowledge for mobile apps. These solutions should also help solve expected dependencies of the model from the context of the organization, e.g., by realizing organization-wide dynamic bases of knowledge refined by domain (mobile health, mobile education, etc.).

10 Previous Works

A basic book concerning Mobile was authored by Brian Fling and published on 2009 by O'Reilly Media [6]. This book includes the fundamental concepts and techniques for creating mobile sites and apps.

On market side, in the year 2010, the Vision Mobile web site examined the latest insights in the mobile market, provided views about the winners and losers of the platform and handset race for 2011, and discussed the challenges facing mobile network operators in their quest to stay relevant to mobile application developers [8].

On the Software Engineering side, in 2010, Tony Wasserman analyzed the issues for mobile development, and provided a view about mobile development environments [9]; moreover, Jeff Rowberg proposed a comparison between different approaches to, and technologies for, mobile development [10].

Subsequently, a comparison between native, web, and hybrid mobile app development approaches was provided by WorkLight Webinar Series [7].

Many other previous studies are concerned with specific platforms, platforms vs. technologies, and development approaches, including: HTML5 versus Android: apps or web [11], Dojo 1.6 [12], PhoneGap to build native apps [13], Appcelerator vs. PhoneGap vs. Adobe Air [14], PhoneGap vs. Flex vs. Appcelerator vs. Corona [15].

Some other studies are concerned with how to correctly exploit technologies, e.g. how to use PhoneGap and the Dojo Toolkit for developing hybrid mobile applications [14].

This paper tried, in its turn, to put together competencies from academic research and a world major company for doing a step ahead in the support of decision making in the mobile apps domain.

11 Conclusions and Future Work

This paper presented a model and guidelines for supporting the choice of the right technological stack for the development of a new mobile application from given

requirements. The proposed model takes into account non-functional requirements, app features, and available development technologies, platforms and approaches. The model is not exhaustive with respect to the technology dimensions and needs; however, it is very simple to utilize, augment and improve by adding new items and attributes. The twelve needs taken into account in this paper are the ones we deem mandatory to consider when making decisions, but their enrichment could boost both quality and type of results.

The method has been empirically tested and further developed by four case studies at the IBM Italy Rome Smart Solutions Lab. These case studies were concerned with different problems to address and covered a wide range of possibilities. However, the number and types of case studies that we utilized to gain experience and eventually populate our base of knowledge (i.e., a simple table) are still limited. Consequently, the items presented in Ordinal scale by Table 1 should not be intended as conclusive values.

Next steps of this study include the development of a larger mix of case-studies, and the study of dependences, if any, of the model on the application domain. One more study we aim to conduct is a deep sensitivity analysis to assess the model robustness with respect to different features and inputs.

Moreover, an extension of the model is planned, which takes in account the value and risk of technologies and processes, as well as the in-house skills and experience.

Furthermore, a theoretical investigation aimed to fully formalize our multi-dimensional space and its mathematical foundation is in our plan for the future.

Last but not least, we aim to create a web site, to be available to the community, where practitioners can share their usage experiences with our framework. This way, it will be possible to gather relevant information from a high number of cases, contexts, and developers. Such information will be used to improve the proposed model and could lead to a new proposal of development process in the domain of the mobile apps.

References

1. Keeney, R.L., Raiffa, H.: Decision with Multiple Objectives: Preferences and Value Tradeoffs. University of Cambridge (1976)
2. Falessi, D., Cantone, G., Kazman, R., Krutchen: Decision-making Techniques for Software Architecture Design: A Comparative Survey. ACM Computing Surveys 43(4) (2011)
3. Zelkowitz, M.V., Wallace, D.R.: Experimental Models for Validating Technology. IEEE Computer 31(5) (1998)
4. Basili, V.R., Weiss, D.: A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering 10(6) (1984)
5. Wohlin, C., Runeson, P., Horst, M., Ohlson, M.C., Regnel, B., Wesslen, A.: Experimentation in Software Engineering. An introduction. Kluwer Academic Publishers (2000)
6. Fling, B.: Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps, pp. 13–27. O'Reilly Media (2009) ISBN 978-0-596-15544-5
7. WorkLight Webinar Series: Native Web or Hybrid Mobile App Development (2011), `http://www.worklight.com/assets/files/Native-Web-Hybrid-Mobile-App-Dev-Webinar.pdf` (accessed on September 10, 2011)

8. VisionMobile: Developer Economics 2010 and Beyonds (2010), http://www.visionmobile.com/blog/2010/07/developer-economics-2010-the-role-of-networks-in-a-developer-world/ (last access August 12, 2011)
9. Wasserman, A.I.: Software engineering issues for mobile application development. In: ACM SIGSOFT FoSER (2010), http://www.cmu.edu/silicon-valley/wmse/wasserman-foser2010.pdf
10. Rowberg, J.: Comparison: App Inventor, DroidDraw, Rhomobile, PhoneGap, Appcelerator, WebView, and AML (2010), http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml (accessed on January 03, 2012)
11. Google I/O: HTML5 versus Android: Apps or Web for Mobile Development? (2011), http://www.google.com/events/io/2011/sessions/html5-versus-android-apps-or-web-for-mobile-development.html (last access on November 15, 2011), http://www.youtube.com/watch?v=4f2Zky_YyyQ (last access August 14, 2012)
12. Lennon, J.: Get started with Dojo Mobile 1.6 - And get a peek at new features coming in 1.7, IBM developerWorks (2011), http://www.ibm.com/developerworks/web/library/wa-dojomobile/index.html (last access December 13, 2011)
13. Yao, M.: What are the pros and cons of using PhoneGap to build native apps? (2011), http://www.quora.com/What-are-the-pros-and-cons-of-using-PhoneGap-to-build-native-apps (accessed on January 03, 2012)
14. Lukasavage, T.: Review: Appcelerator vs. PhoneGap vs. Adobe Air (2011), http://savagelook.com/blog/portfolio/appcelerator-vs-phonegap-vs-adobe-air (accessed on December 28, 2011)
15. Vilches, A.: PhoneGap vs. Flex vs. Appcelerator vs. Corona: nuevas conclusiones (2011), http://www.xydo.com/toolbar/27165658-phonegap_vs_flex_vs_appcelerator_vs_corona_nuevas_conclusiones_%C2%AB_yo_programador (accessed on January 03, 2012)
16. Dingsor, A.: Writing a Hybrid Mobile Apllication with PhoneGap and the Dojo Toolkit. IBM (2011), http://public.dhe.ibm.com/software/dw/web2mobile/07072011_dingsor/hellohybrid.pdf (accessed on January 04, 2012)
17. Cantone, G., Donzelli, P.: Software Measurements: from Concepts to Production, T.R. Intl. Software Engineering research Network, ISERN T.R. 97-27 (1997) (In Italian)
18. Cantone, G., Donzelli, P., Pesce, G.: Misure software: teoria, modelli e ciclo di vita, in Metriche per il. Software, Ed. GUFPI-ISMA, Franco Angeli (2006) (In Italian)
19. Mastrofini, M., Cantone, G., Shull, F., Diep, M., Seaman, C., Falessi, D.: Enhancing the System Development Process Performance: a Value-Based Approach. In: Procs. of INCOSE 2012, Rome, Italy (2012)
20. Kirby, R.C., Siebenmann, L.C.: Foundational Essays on Topological Manifolds. Smoothings, and Triangulations. Princeton University Press (1977)
21. Boehm, B.W.: Value-based software engineering: Overview and agenda. Tech report, USC-CSE-2005-504, University of Southern California, Park Campus. Los Angeles, CA, USA (2005)
22. Basili, V.R.: Quantitative evaluation of software engineering methodology. In: Proc. First Pan Pacific Computer Conf., Melbourne, Australia, September 10-13 (1985)
23. Cantone, G., Donzelli, P.: Production and Maintenance of Software Measurement Models. Journal of Software Engineering and Knowledge Engineering 5 (1998)

Table 1. The graphical view of an instance of our model for some actual needs, device features, tecnologies, and platforms

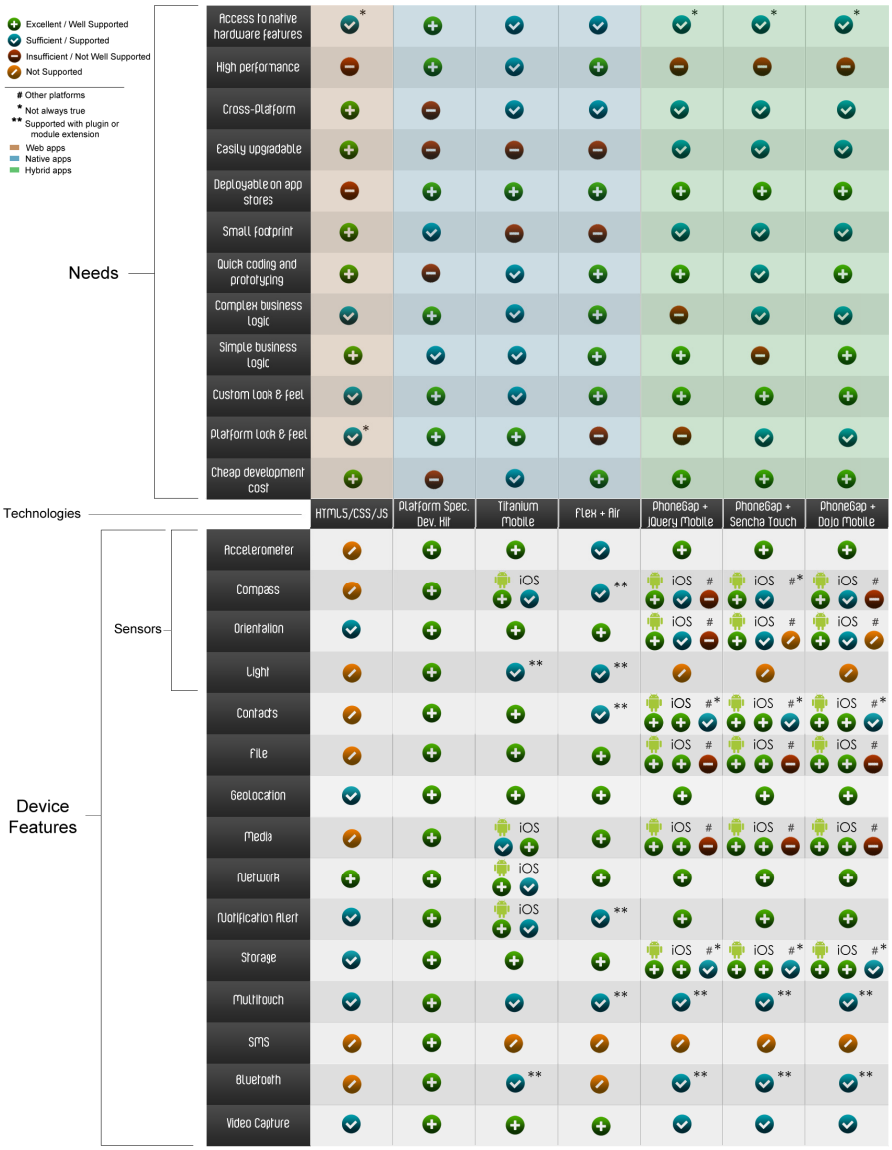

Table 2. Requirements of a Case-study (simplified version)

Description	The app is required to allow an user to chat with other people connected to network by the same application. Moreover, the app is required to provide the user with the ability of filing and sending recorded audio clips and video messages. Furthermore, the app is required to allow an user to extend her/his contact list by specifying the telephone number of the person s/he wants to include; her/his mobile phone list of contacts is expected to be allowed as a source for such a phone number.	
Requirements	1. Available on Android and iOS 2. Access to media 3. Access to videocamera 4. Access to local files 5. Access to contacts list	6. Notification Alert 7. High performance 8. Deployable on app stores 9. Cheap development costs 10. Access to network

Table 3. Results from the Case-study