



Evaluation Criteria for Mobile Functional Prototyping

Trabalho de Conclusão de Curso

Engenharia da Computação

Nome do Aluno: Paula Beserra Pithon

Orientador: Prof. Joabe Bezerra de Jesus Júnior



UNIVERSIDADE
DE PERNAMBUCO

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

Paula Beserra Pithon

EVALUATION CRITERIAS FOR MOBILE FUNCTIONAL PROTOTYPING

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2019.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 11/6/2019, às 21h, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **PAULA BESERRA PITHON**, orientado(a) pelo(a) professor(a) **JOABE BEZERRA DE JESUS JÚNIOR**, sob título **Functional Prototyping Frameworks for Mobile Development**, a banca composta pelos professores:

JOÃO HENRIQUE CORREIA PIMENTEL

JOABE BEZERRA DE JESUS JÚNIOR (ORIENTADOR) (PRESIDENTE)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

Aprovada Aprovada com Restrições* Reprovada

e foi-lhe atribuída nota: 10,0 (dez)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.



AVALIADOR 1: Prof (a) **JOÃO HENRIQUE CORREIA PIMENTEL**



AVALIADOR 2: Prof (a) **JOABE BEZERRA DE JESUS JÚNIOR**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final

À minha família, marido e todos que apoiaram esta jornada.

Agradecimentos

Agradeço primeiramente a UPE pela oportunidade de estar inserida em uma das universidades mais prestigiadas do país. Agradeço também a todos os professores que cruzaram meu caminho, pelos ensinamentos e oportunidade de crescer academicamente em cada matéria que aprendi. Também agradeço as empresas que estive inserida nesses últimos cinco anos no qual pude aplicar todos os conceitos aprendidos na prática; Manifesto Games e CESAR.

Agradeço imensamente a minha família, especialmente meus pais, que esteve comigo me apoiando por todos esses anos, sendo sempre compreensivos e motivadores. Agradeço também a meu namorado, Pedro, no qual não apenas me apoiou, como também me auxiliou no processo de escrita deste TCC desde o começo. Também agradeço a meus amigos que acompanharam esta jornada desde o começo no qual estive junto em todos os semestres; Maria Paula, Thainá e Gabriel.

A todos que estão tirando um tempo para ler esse trabalho meu muito obrigada. Este TCC é para todos vocês!

Resumo

Este estudo tem como objetivo compreender as práticas atuais de prototipagem no desenvolvimento de aplicações móveis e realizar uma análise dos critérios de escolha utilizados atualmente pelos desenvolvedores. Para construir um aplicativo, a prototipação é um passo essencial, afinal, reduz o risco do projeto e aumenta a chance de sucesso. Existem duas principais metodologias de prototipagem usadas atualmente; Protótipo Descartável, no qual o código da aplicação é descartado, e Protótipo Evolutivo, no qual o projeto prototipado evolui para o produto final. Como cada metodologia é usada em diferentes aplicativos, definir uma estrutura de desenvolvimento móvel pode ser uma tarefa complexa para desenvolvedores iniciantes, portanto, é importante entender o que faz com que engenheiros experientes escolham frameworks específicos ao criar protótipos para plataformas móveis. Devido à natureza deste objetivo de pesquisa, optamos por conduzir e analisar uma pesquisa usando a estrutura de sete estágios do Kasunic. A fim de reunir as informações necessárias para comparar essas metodologias de prototipagem definimos 8 critérios de escolha comumente usados da literatura. O questionário foi construído em duas seções; a primeira consistiu em questões referentes à demografia da pesquisa e a segunda incorporou questões quantitativas e qualitativas em relação a esses critérios pré-definidos. Distribuimos a pesquisa entre os desenvolvedores de aplicativos móveis especializados em Recife, no Brasil, por meio de plataformas populares e acabamos com mais de 50 respostas de todo o país. Terminamos com uma análise detalhada dos dados sobre os critérios utilizados pelos entrevistados e uma discussão com pontos de melhoria para o levantamento distribuído.

Abstract

This study aims to gain an understanding of the current prototyping practices in mobile app development and conduct an analysis of choice criteria currently used by developers. In order to build an app, prototyping is an essential step, after all, it reduces project risk and increases the likelihood of success of any software. There are two main prototyping methodologies currently used; throw-away prototype, in which the code of the application is discarded, and evolutionary prototype, in which the prototyped project evolves into the final product. Since each prototyping methodology is used in different applications, defining a mobile development framework might be a complex task for beginner developers, therefore it is important to understand what makes experienced engineers chose a specific development frameworks when prototyping for mobile platforms. Due to the nature of this research goal, we chose to conduct and analyse a survey using Kasunic's seven stage framework. In order to gather the information we needed to compare these prototyping methodologies, we defined 8 commonly used choice criterias from literature. The questionnaire was built in two sections; the first consisted of questions regarding the demographics of the research and the second incorporated quantitative and qualitative questions regarding these predefined criterias. We distributed the survey among expert mobile developers in Recife, Brazil, through popular platforms and ended up with over 50 responses from all around the country. We ended up with a detailed data analysis regarding criterias used by respondents and a discussion with improvement points for the distributed survey.

Summary

List of Figures	10
List of Tables	12
List of Symbols and Abbreviations	13
Chapter 1 Introduction	14
1.1 Motivation	
1.2 Goals	16
1.3 Document Structure	17
Chapter 2 Theoretical Background	19
2.1 Concepts	19
2.1.1 Mobile Development	21
2.1.2 Functional Prototyping	21
2.2 Related Work	22
Chapter 3 Survey Methodology	24
3.1 Modeling	
3.2 Demographics of the Research	25
3.3 Building the Questionnaire	25
3.3.1 Evaluation Criteria	26
3.3.2 Questions	27
3.4 Release and Analysis	28
Chapter 4 Results	30
4.1 Respondents Overview	32
4.2 Data Analysis	33
4.2.1 Throw-away Prototype Data	33
4.2.2 Evolutionary Prototype Data	35
4.3 Discussion	37
Chapter 5 Conclusion	39
3.1 Future Work	39
References	41
A. Survey Questions	44

List of Figures

Figure 1. Mobile apps download growth from 2009 to 2017.	16
Figure 2. Comparison between Smartphone usage and Desktop usage.	21
Figure 3. Subdivisions in three large groups of mobile app development.	22
Figure 4. Steps of the methodological process chosen for this research.	26
Figure 5. Screenshot of the publicly available survey.	27
Figure 6. Example of a survey using a one-stage Likert Scale.	29
Figure 7. Our dashboard report using Google Data Studio.	30
Figure 8. Number of respondents from each Brazilian state.	31
Figure 9. Respondents' software development experience time.	32
Figure 10. Respondents' mobile development experience time.	32
Figure 11. Technology expertise of the respondents.	33
Figure 12. Bar chart that represents the scale regarding throw-away prototype criterias.	35
Figure 13. Bar chart that represents the scale regarding evolutionary prototype criterias.	37

List of Tables

Table 1. Representation of final results and criteria evaluation.

38

List of Symbols and Abbreviations

GPS – Global Positioning System

HTML – Hypertext Markup Language

CSS – Cascade Style Sheets

MVPs – Minimum Viable Product

IEEE – Institute of Electrical and Electronics Engineers

API – Application Program Interface

BPMN – Business Process Model Notation

OS – Operating System

CPU – Central Processing Unit

CEOs – Chief Executive Officer

CTOs – Chief Technology Officer

POC – Proof of Concepts

AI – Artificial Intelligence

UX – User Experience

Chapter 1

Introduction

This chapter is devoted to the presentation of this graduation thesis. Initially, we will explain the motivation for which the proposed theme was chosen, then, we will describe the general and specific objectives outlined for this project and, finally, we will conclude detailing the structure of the following chapters.

1.1 Motivation

Mobile phones combining a range of different functions such as media player, camera, and GPS (Global Positioning System) with advanced computing abilities and touchscreens, alias Smartphones, are enjoying ever-increasing popularity. They enable innovative mobile information systems known as mobile applications, often referred to as *apps* [1].

While application development for mobile devices goes back at least 15 years, there has been exponential growth in mobile application development since the Apple's iPhone App Store opened in July, 2008 [2]. There are currently around 5,000,000 apps on both Apple's App Store and Android's Google Play [3] [4] generating \$1.3 trillion in revenue on 2016 and making it one of the fastest growing markets around with over 3.5 billion users [5] as seen on **Figure 1**.

Since the market of mobile operating systems for smartphones is fragmented and rapidly changing [1] all platforms differ significantly from each other. In this scenario, it is possible to divide mobile app development into three categories: native, web-based, and hybrid. Native applications run on a device's operating system and are required to be adapted for different devices, web-based apps require a web browser on a mobile device and hybrid apps are 'native-wrapped' WebApps [1] [6] [7].

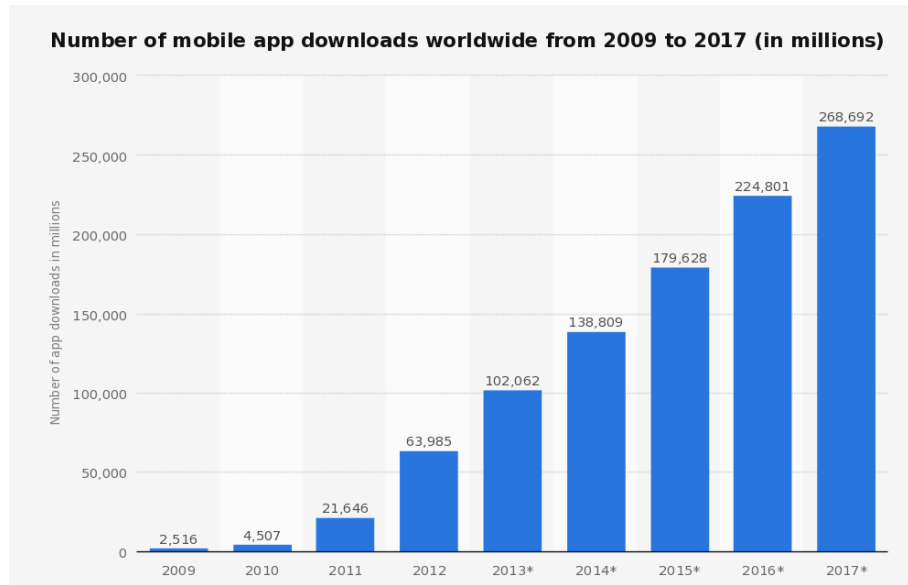


Figure 1. Mobile apps download growth from 2009 to 2017 [5].

Whenever software developers start to build any kind of application, prototyping is a meaningful step; In any situation in which a system must be developed from scratch, the conception itself should start in a prototyping phase. It is well understood that undetected errors that occur in the requirements phase of system development are the most costly to repair in later stages [8].

The main reason for prototyping is to gather knowledge and increase the likelihood of success of a software [9] [10]. More generally, prototyping may also be considered as a way of reducing the project risk that comes from incomplete knowledge of what is required or how to achieve it [10] and facilitating the requirements phase for any type of software [9].

There are two prototyping methodologies that are broadly used: throw-away prototype, that involves building the software for testing or validation purpose when its main features are not yet clear, and evolutionary prototype, in which the software is built considering an architecture and features that will be reused in the future in order to develop the full product [11].

Currently, open source projects, technical blogs and socio-professional media are perceived as the 'key information resource' for software development [12]. The

ability to search, understand, and use this online knowledge is one of the key abilities affecting software engineers' efficiency and success and it is something acquired with time and not accessible to beginner or inexperienced developers [13]. Since the information contained on these resources are too broad, a comparative analysis becomes necessary to novice software engineers in a short-term project or prototypes, in order for them to accomplish their goals regarding technologies they have never engaged with previously [14].

1.2 Goals

This study aims to gain an understanding of the current prototyping practices in mobile app development and conduct an analysis of mobile frameworks categories currently used by developers.

The idea behind this comparative study in form of a guideline for young or beginner software engineers is to gather knowledge from more experienced developers in order to create a clearer path to follow when learning a new technology. That becomes especially useful when building prototypes, since these projects often have shorter time to be executed and cannot have a steep learning curve, even with its complexity [11] [15].

In order to accomplish this goal, we aim to:

- Present concepts revolving prototyping and mobile development, as well as group them into pertinent categories to be analysed;
- Establish relevant comparison criteria that are considered when building either a throw-away prototype or an evolutionary prototype.
- Develop and publish a survey directed to senior software engineers in order to determine which criterias are more relevant when choosing one of two prototyping methodologies;
- Conduct an analysis of the answers comparing current mobile frameworks available;

- Determine which of these frameworks are more compatible to each prototyping methodology.

1.3 Document Structure

This document is divided into five chapters: Introduction, Theoretical Background, Methodology, Results and Conclusion. We describe each of these sections below:

Chapter 1 - Introduction: This chapter gives an overview of the context in which this research is inserted. It also points out the motivations in why this particular theme was chosen, as well as describe the general objective and specific goals of the entire thesis.

Chapter 2 - Theoretical Background: This chapter is the theoretical base to the concepts mentioned over this entire document. Initially we describe pertinent concepts regarding mobile development and prototyping methodologies. Next, we list all related works that served as an inspiration to this research.

Chapter 3 - Methodology: This chapter describes all steps of the methodological process used in this research. First, we describe the seven stage framework used in order to produce the survey created for this research. Next, we describe all steps regarding the demographics of the research, the construction of the questionnaire and finally, the data analysis and tools used.

Chapter 4 - Results: This chapter presents the results regarding the survey distributed. On the initial subsection we show the demographics of the respondents, such as location, experience with software engineering and technology expertise. Next, we present the results regarding the quantitative and qualitative questions when comparing evaluation criteria among prototyping methodologies. Finally, we discuss the findings and compare each mobile framework when considering either of the methodologies described.

Chapter 5 - Conclusion: This chapter presents the final conclusions of the proposed model, the limitations found in its execution and possible future work regarding the research.

Chapter 2

Theoretical Background

The goal of this section is to present concepts regarding mobile development frameworks for functional prototyping, as well as related scientific works explaining their proposed approaches and its relevance.

2.1 Concepts

In order to gather a better understanding of this research, it is important to highlight two concepts that were mentioned previously and the division of their categories. Initially we are going to discuss the different groups regarding mobile development and then, categories that revolve around functional prototyping methodologies that will be mentioned in the following sections.

2.1.1 Mobile Development

In many ways, developing mobile applications is similar to software engineering for other embedded applications. However, mobile applications present some requirements that are less commonly found with traditional hardware applications, including: Potential interaction with other applications, sensor handling, 'closed' security, intuitive user interfaces, optimized processing, etc [2]. These requirements are common in mobile devices currently on the market and are seen as positives by users when compared to other platforms [16] as shown on **Figure 2**.

In order to better analyse the mobile development tools available, we chose to categorize them into three different groups. Different frameworks produce different outcomes but can be broadly grouped into: native development, web-based development, and hybrid development [7]. Each approach carries inherent benefits and limitations, and finding the one that best addresses the organization's needs could be a challenging task [17].

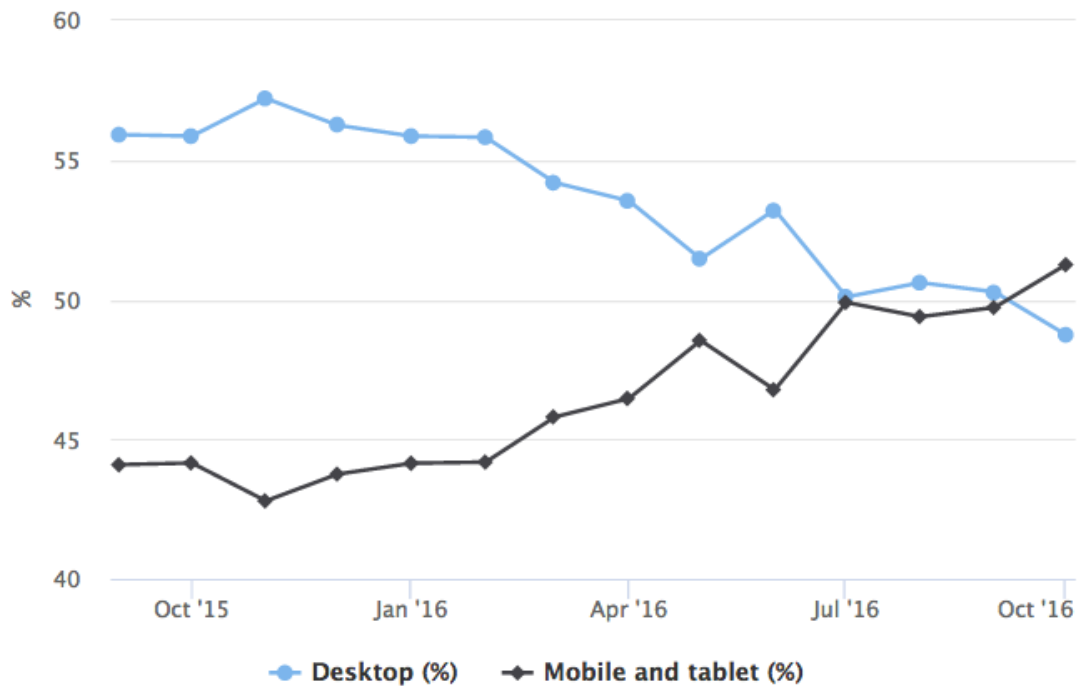


Figure 2. Comparison between smartphone usage and desktop usage [16].

Native applications run on a device's operating system and are required to be adapted for different devices such as Android and iOS. Web-based apps require a powerful web browser on a mobile device that supports HTML (Hypertext Markup Language), CSS (Cascade Style Sheets) and Javascript. The hybrid development approach combines native development with web technology, such as Ionic, Cordova and PhoneGap [12] [13] [17]. Even though we chose to focus on these three divisions, there are sub-categories in each of these large groups. Each subdivision represents small architectural changes on each largest mobile development group that can be seen on **Figure 3**.

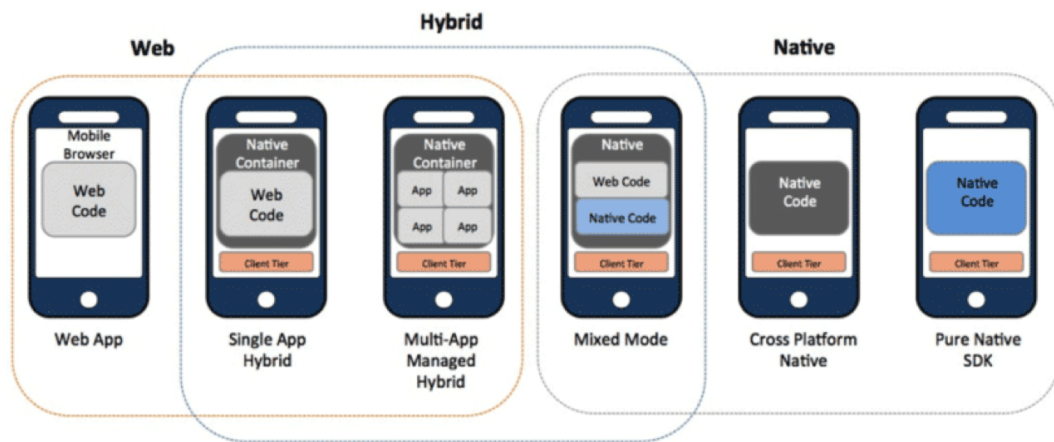


Figure 3. Subdivisions in three large groups of mobile app development [17].

2.1.2 2Functional Prototyping

Prototyping is the process of developing a trial version of a system (a prototype) or its components or characteristics in order to clarify the requirements of the system or to reveal critical design considerations [18]. The primary reason for prototyping is to acquire knowledge and thus reduce uncertainty and increase the likelihood of success of a software project and is usually necessary in situations where it is not known precisely what to build nor, in some cases, how to build it [10].

Selecting an appropriate development approach is crucial to building a successful software system [11]. Prototypes can be developed either to be thrown away after producing some insight or to evolve into the product version. Each of these approaches has its benefits and disadvantages and the most appropriate choice depends on the context of the effort [9].

The throw-away approach is most appropriate in the project acquisition phase where the prototype is used to demonstrate the feasibility of a new concept and to convince a potential sponsor to fund a proposed development project. The evolutionary approach produces a series of prototypes in which the final version becomes the software product. This approach depends on special tools and

techniques because it is usually not possible to evolve a prototype into production use without significant changes to its implementation [9].

Both throw-away and evolutionary prototypes are used in many scenarios, such as enterprise applications, academic projects, military plans [11], startup MVPs (Minimum Viable Product) [19] as well as hackathons [20] and other short term competitions. Although an experienced team is the best approach to guarantee a project's success, sometimes it is necessary to start fresh in a technology that was not yet mastered since software development often requires knowledge beyond what developers already possess [12].

2.2 Related Work

Beynon-Davies et al. study shows prototyping has been discussed since the late-1970s in the information system development literature, presenting the relevance of this concept since the early software engineering days [21]. Their paper covers not only the discussion revolving prototyping and its early processes, but also explores the history of prototyping, and previous methodologies, such as pioneer waterfall model.

The '11th IEEE (Institute of Electrical and Electronics Engineers) International Workshop on Rapid System Prototyping' contains an explanatory article about rapid system prototyping [9] as it is known today and its methodologies. Kordon et al. explains the main reason for using prototypes: prototype versions of most systems are much less expensive to build than the final versions. In their research, readers also have a grasp on prototyping main approaches; when they are developed either to be thrown away, after producing some insight, or to evolve into the product version. Those approaches can also be seen on Gordons et al. research [11], in which they categorize all study cases in either one of two methodologies: throw-away or evolutionary;

Each of these approaches mentioned before has its benefits and disadvantages and the most appropriate choice depends on the context of the effort. Gordons et al. research uses commonalities among published case studies of rapid

prototyping in order to develop a guideline on how to use rapid prototyping effectively, as we pay special attention to factors that contribute to the selection of one prototyping method over another. Their research also shows the product attributes commonly affected by prototyping and the results of all case studies related to these attributes, such as ease of use, performance, design features, maintainability and others that become a base to our work.

Heitkotter et al., also consolidates a list of 14 relevant criteria when comparing different cross-platform development approaches [1], that were used when choosing which we were going to use in this research, as well as Palmieri's study that also raises important criterias when comparing mobile development approaches such as programming languages, availability of APIs (Application Program Interface), etc [22]. Joorabchi et al. [7] conducts a grounded theory approach to real challenges in mobile development, that does not contain specific criterias when comparing the results, but rather concepts identified during the research.

Palmieri's also does a terrific job listing cross-platform frameworks and enumerating its main advantages over native mobile development approaches. Jobe [23] on the other hand, not only clearly explains the difference between native and mobile WebApps, but he also compares both categories of mobile development as well as refers practical studies that supports its theoretical basis.

Chapter 3

Survey Methodology

Considering the nature of this research goal, we started by conducting and analyzing a survey with senior software developers, who are experts in either native app development, WebApp development or hybrid development for both iOS and Android platforms, followed by a discussion and a comparative study among prevailing development technologies.

A survey is a method to collect and summarize evidence from a large representative sample of the overall population of interest. In software engineering, surveys are one of the most frequently used research methods for conducting empirical investigation studies [24]. There are many studies in building an effective survey in software engineering [25] and, in order to design and conduct a survey research, we decided on the seven stage framework proposed by Kasunic [24].

3.1 Process Modeling

To better understand this process, we chose to model it in a graphic representation form. We modeled the tasks according to Kasunic's framework using BPMN (Business Process Model Notation) version 2.0 [26], since it is easy to diagram and understand due to its objectivity in representation [27]. Its final model can be seen on **Figure 4**.

In order to facilitate the organization of this research, we will divide each of these steps in sub chapters. The initial subsection will expose the demographics of the research, in which we will discuss the first and second steps of the chosen framework. The next subsection will discuss the fourth step regarding the construction of the survey itself. The final subsection will explain the fifth, sixth and seven steps of Kasunic's methodology, in which we disclose the final steps of the research such as the release of the survey and analysis of the results.

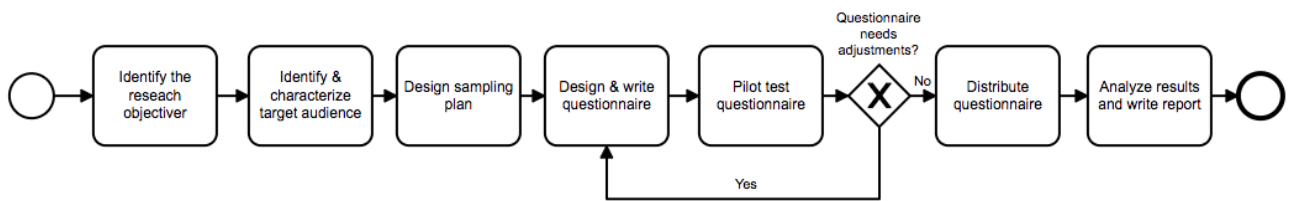


Figure 4. Steps of the methodological process based on Kasunic's framework [Own authorship].

3.2 Demographics of the Research

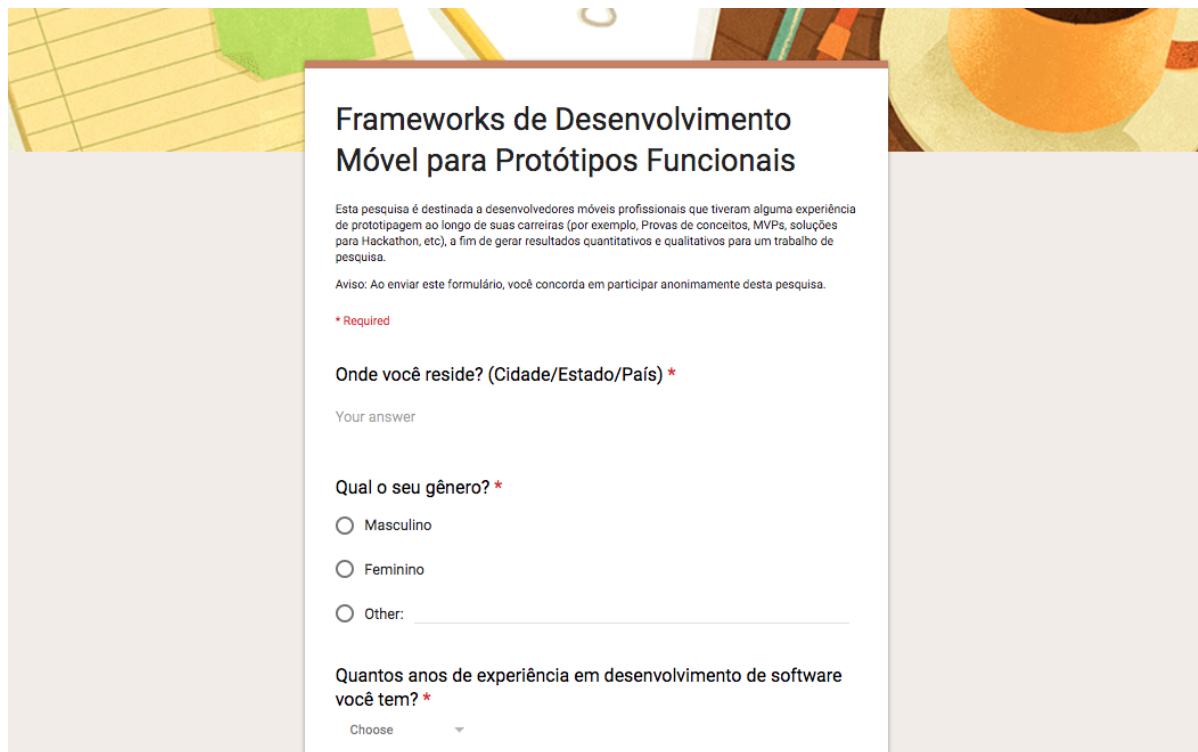
The first step is to define our research objective which was discussed initially in the previous section. After initially defining the goal of our research, the second step is to identify and characterize the target audience of this research. We chose the audience as being senior software engineers, characterized by having at least 5 years of experience as a developer [28]. It was also important to consider developers who had at least intermediate knowledge in at least two mobile technology categories in order to gather a more generalist approach on the subject.

The next step is to design the sampling plan. We determined that the survey would be released and disclosed mainly to software engineers residents of Recife, Brazil, but also open to other regions of the country. That was decided based on having a relevant sample size to our research and the location of where the survey is being conducted, as well as guarantee the quality of the answers since part of the answers are discursive.

3.3 Building the Questionnaire

The fourth step is to design and write the questionnaire. The survey objectives and internal questions were translated into a portuguese carefully-worded questionnaire and designed to facilitate analysis and interpretation. The first section of the survey contains questions regarding the demographics of the respondents,

including the location in which the respondent resides, mobile technologies known as well as years of experience as a developer. After this initial section, there is the qualitative and quantitative research divided by prototyping methodology and each evaluation criteria that will be discussed below.



The screenshot shows a survey form with the following content:

Frameworks de Desenvolvimento Móvel para Protótipos Funcionais

Esta pesquisa é destinada a desenvolvedores móveis profissionais que tiveram alguma experiência de prototipagem ao longo de suas carreiras (por exemplo, Provas de conceitos, MVPs, soluções para Hackathon, etc), a fim de gerar resultados quantitativos e qualitativos para um trabalho de pesquisa.

Aviso: Ao enviar este formulário, você concorda em participar anonimamente desta pesquisa.

* Required

Onde você reside? (Cidade/Estado/País) *

Your answer

Qual o seu gênero? *

Masculino

Feminino

Other: _____

Quantos anos de experiência em desenvolvimento de software você tem? *

Choose ▾

Figure 5. Screenshot of the publicly available survey [Own authorship].

3.3.1 Evaluation Criteria

Our main objective is set to better categorize three largest categories for mobile platform development when it comes to functional prototyping based on predetermined criteria. For that matter, it is important to list each evaluation criteria throughout the survey and match it with one of the two prototyping methodologies we defined, alongside a justification from each professional interviewed based on their professional experience. The evaluation criterias are:

1. **Use of a know programming language** and the knowledge curve to acquire knowledge on this framework.
2. **Technology community size** and maturity of the framework as well as resources available online to help initially.
3. **Development and build time for each OS (Operating System)**, from creating the project to actually compiling and building it on the platform.
4. **Fidelity to final designed product** and how easily it is to develop interfaces designed previously before the functional prototype
5. **Native user experience for mobile users** and how close the application experience approaches a native experience.
6. **Maintainability** which is the amount of resources to implement new features as requested and fix bugs while on production.
7. **App performance** including CPU (Central Processing Unit) usage, application loading time on user device, etc.
8. **Third party APIs availability** of free or open source APIs to reduce development time.

3.3.2 Questions

The questions regarding the criteria were designed using the one-stage Likert Scale [29] [30] in which the respondents specify their level of agreement or disagreement on a symmetric scale from 'not important' to 'very important' for a series of statements while responding in each of the prototyping methodologies context, such as **Figure 6**. As for the qualitative questions, they are discursive so the respondent can comment on any of the criterias available, explain his motivation to choose such a level of agreement as well as, in the end, suggest and justify the addition of more relevant criterias. The final questions in this survey can be seen on **Appendix A**.

In a scenario in which you have to build a throwaway prototype (i.e. to prove a concept or validate a specific feature) and the code will not be used afterwards, how relevant do you consider each of these criterias when choosing a framework? *

	Not important	Indiferent	Very important
Required skills to develop applications due to a known programming languages	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Technology community size	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Development and build time for each OS (from creating a project to running it on a platform)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 6. Example of questions using a one-stage Likert Scale [Own authorship].

3.4 Release and Analysis

The fifth step in Kasunic’s framework is to pilot test the questionnaire. It is important to ask external experts on mobile development to review the survey in order to make sure all questions are appropriate and easily comprehensible. The survey presented itself as being understandable in general, but based on feedback, it was necessary to return to the previous step and change the initial criteria before the final pilot test was a success.

The sixth step of the chosen framework is to distribute the questionnaire, this survey was advertised and made publicly available for one month, being constantly advertised in social media and relevant platforms for senior software developers in Brazil, as well as sent directly to professionals that identify as having more than 5 years of experience in mobile development on LinkedIn, that is currently the largest professional network available [31].

The last step on our model is to analyze the results and write the final report. In order to analyze the results collected and translate them into appropriate graphical

displays that facilitate understanding, we used Google Data Studio [32], a free to use web-based analytics tool. With Google Data Studio, it is possible to convert our acquired quantitative data into appealing and informative reports containing charts and graphs such as **Figure 7**.

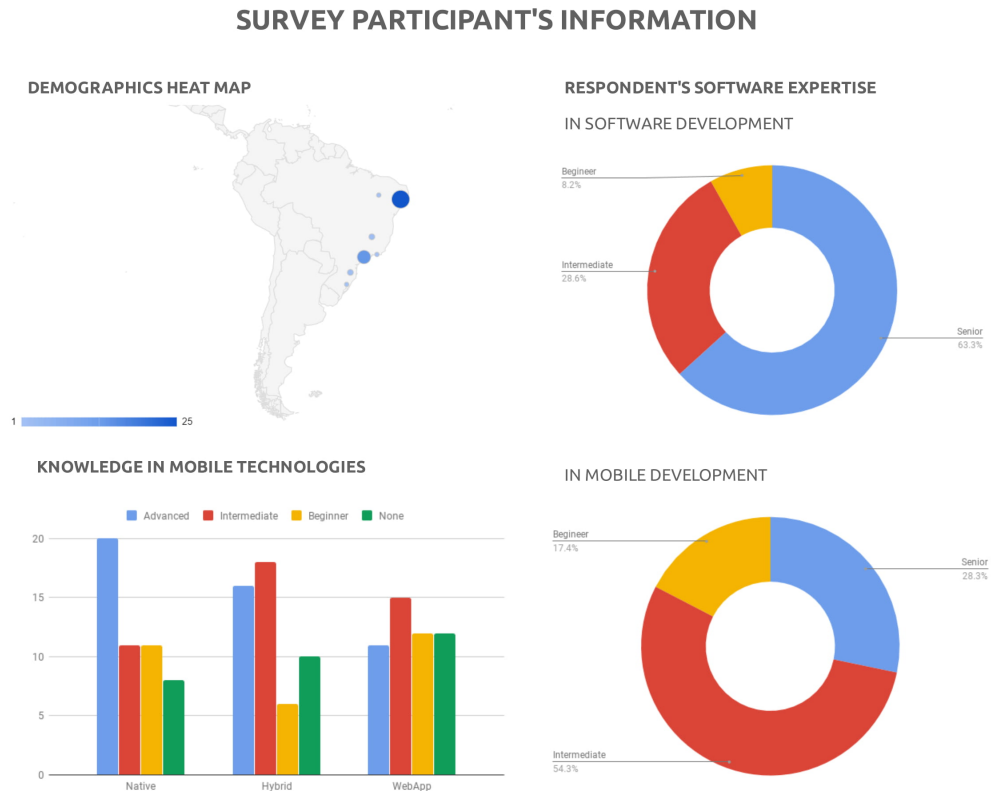


Figure 7. Our dashboard report using Google Data Studio [Own authorship].

The combination of the initial research and the semi-structured survey resulted in a detailed table that listed which evaluation criteria was more relevant for each type of prototyping methodology. This outcome made our discussion regarding mobile development framework categories and prototyping methodologies more consistent and gave a more solid background based on the experts answers. The final result will be discussed in the following section.

Chapter 4

Results

This chapter aims to display the results of the launched survey. Initially we show the demographics of the research, as well as the seniority level of the respondents. Next, we make an analysis of the quantitative and qualitative answers and compare the chosen criterias among prototyping methodologies.

4.1 Respondents Overview

The survey was available for a whole month and was disclosed in different online streams, such as LinkedIn, Facebook and Telegram. During this time, the questionnaire gathered 50 answers from several cities and experience levels. Even though it was initially directed to mobile developers residents of Recife, Brazil, software engineers from all regions of Brazil answered and gave their input as shown in **Figure 8**.

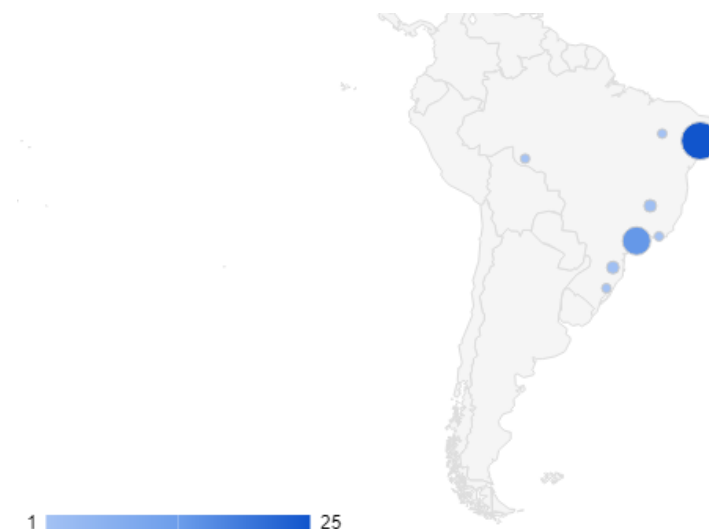


Figure 8. Number of respondents from each Brazilian state [Own authorship].

From all survey respondents 86% had a development role within their company. Other roles included consultants, CEOs (Chief Executive Officer) and CTOs (Chief Technology Officer), all of those had at least intermediate experience in any of the technologies stated. Over 63% characterized as being a senior software developer (**Figure 9**), possessing more than 5 years development experience and 28% of those also had the same amount of experience regarding mobile development (**Figure 10**).

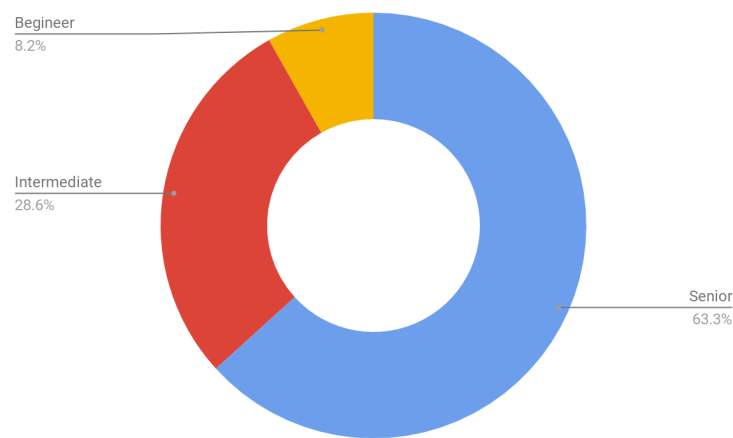


Figure 9. Respondents' software development experience time [Own authorship].

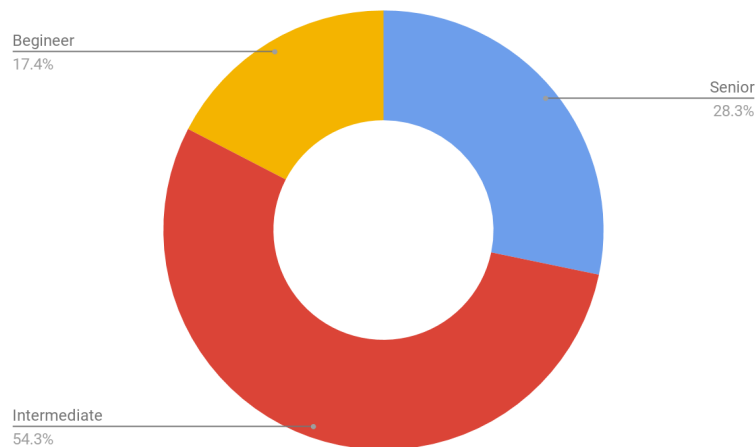


Figure 10. Respondents' mobile development experience time [Own authorship].

A great majority declared having advanced knowledge in native mobile development (either Android or iOS), while hybrid and WebApp development gather less experts on the matter (**Figure 11**). Intermediate knowledge in at least two of the framework groups was an important factor when considering the most relevant answers. Around 60% of the respondents satisfied this condition.

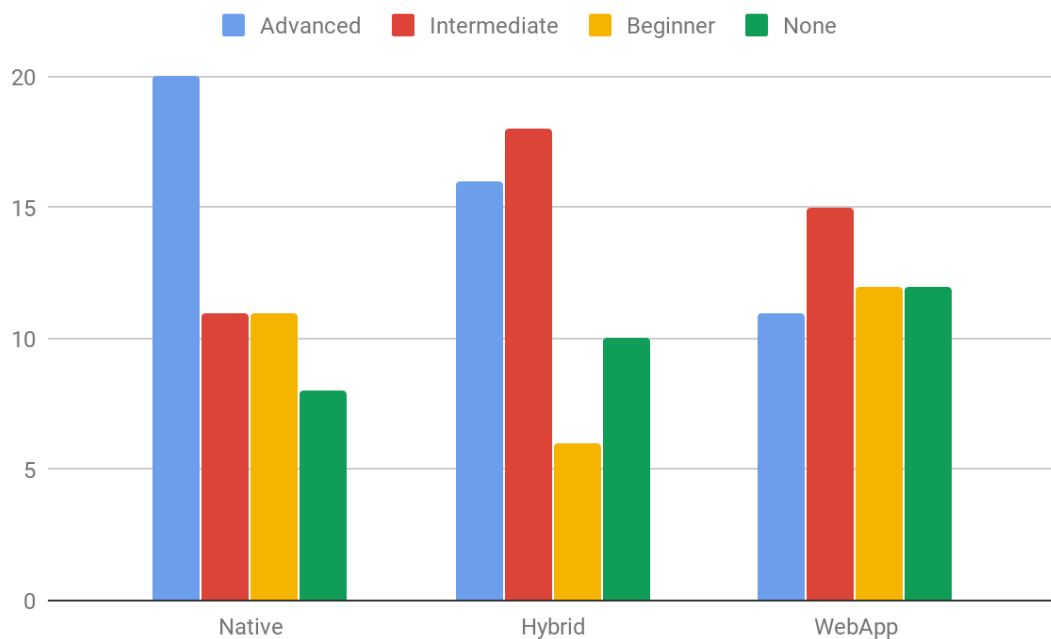


Figure 11. Technology expertise of the respondents [Own authorship].

From all respondents, 86% stated they have had previous experience with prototyping methodologies for mobile development. They were mostly commercial POCs (Proof of Concepts), but other applications included academic projects and MVPs. They were asked to briefly describe their projects and what frameworks and APIs were used.

One interesting fact was that many of the respondents mentioned non-functional prototyping tools when the application developed only required screen navigation. Another interesting insight was to see how AI (Artificial Intelligence) plays a big role on prototyping methodologies; most of the respondents that prototyped an

AI driven system used native development to develop such applications with a wide range of APIs. In general, the survey turned out to demonstrate a distinct range of applications and methodologies experience among respondents, confirming the participation of a diverse public, as we aimed for in this research.

4.2 Data Analysis

In this subsection, we will analyse the answers given based on the Likert Scale, as mentioned in the previous chapter. In order to enhance the visual experiences, we created graphics based on the final score of each evaluation criteria. The scale is represented by integers 1-5 as 1 representing 'not important' and 5 representing 'extremely important' and the final results can be seen on **Figure 12** for throw-away methodology and on **Figure 13** for evolutionary methodology.

4.2.1 Throw-away Prototype Data

It was unanimity among all respondents that 'development and build time for each OS' criteria was important in at least some level compared to others, making it the most relevant for this prototyping methodology, as seen on **Figure 12**. According to the qualitative answers of the participants, it is usually not worth to spend neither money nor time in a system that is going to be discontinued, hence, the need for the application to be developed as fast as possible.

There were three criterias that were between the 'moderately important' and 'very important' zone. 'use of a known programming language' was the second most important criteria; respondents stated that, since throw-away prototypes are usually about testing and implementing a feature that is obscure to the developer, the use of a familiar programming language is preferential and makes the task smoother.

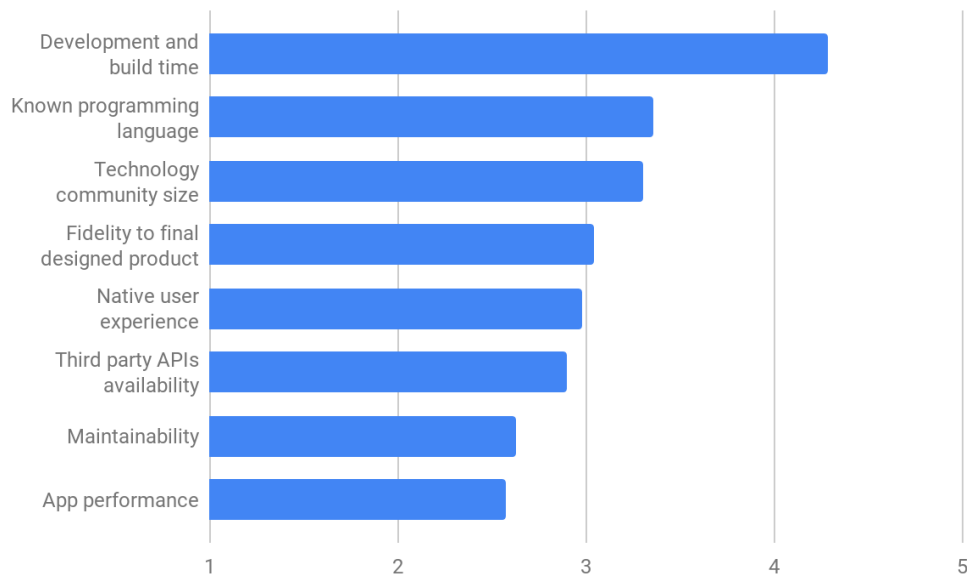


Figure 12. Bar chart that represents the scale regarding throw-away prototype criterias [Own authorship].

The next criterias were 'technology community size' and 'fidelity to final designed product'. Both of these criterias, even though occupying relevant positions on the list, had controversial opinions. A part of the respondents stated that the community size was not important in the long run since many of the features contained within the application did not need to be functional, as another part said that it was important since documented problems could help optimize the development time of the application. Regarding 'fidelity to the final product', it was stated that it is important only if the interface needs validation, otherwise, it should not matter compared to other criterias.

The least relevant criterias were between 'reasonably important' and 'moderately important' marks. 'native user experience' was considered reasonably important if the goal of the prototype is to validate UX (User Experience) like the previous criteria mentioned. Even though the numbers showed that 'third party APIs availability' is not high ranked among the criterias, respondents advocated in favor, since these interfaces drastically decrease development time.

The criterias on the bottom of the list were 'maintainability' and 'app performance'. Since a throw-away prototype is built to be disposable, this criterias did not show relevance according to the respondents since they are not part of a final product. Both 'maintainability' and 'app performance' influence mainly on user retention of market applications. Since a throw-away prototype is not designed to move on this phase, these criterias are not important.

4.2.2 Evolutionary Prototype Data

Evolutionary prototype data showed to be extremely divergent from the previous methodology. While throw-away criterias demonstrated to be cleared defined along the scale, evolutionary criterias were all considered important at some level. For that reason we will be discussing the criterias between 'extremely important' and 'very important' then the criterias that classified between 'reasonably important' and 'very important' as seen on **Figure 13**.

According to the respondents, the most highly ranked criteria was 'maintainability". Since the code developed on the prototype is going to be reused, maintenance costs should be as low as possible since the prototype should always be evolving when adding new features without drastic structural changes. It was also pointed out that spending more time on the beginning of the prototype defining a robust architecture means saving time and money when it grows into the actual product.

The second most important criteria was 'fidelity to final product'. According to respondents, since the app market is extremely competitive, users demand high quality in applications they download and retain on the phones. A bad layout means the brand loses credibility among its users. For that reason, the layout of the prototype has to be as faithful to the designed product as possible since the early stages.

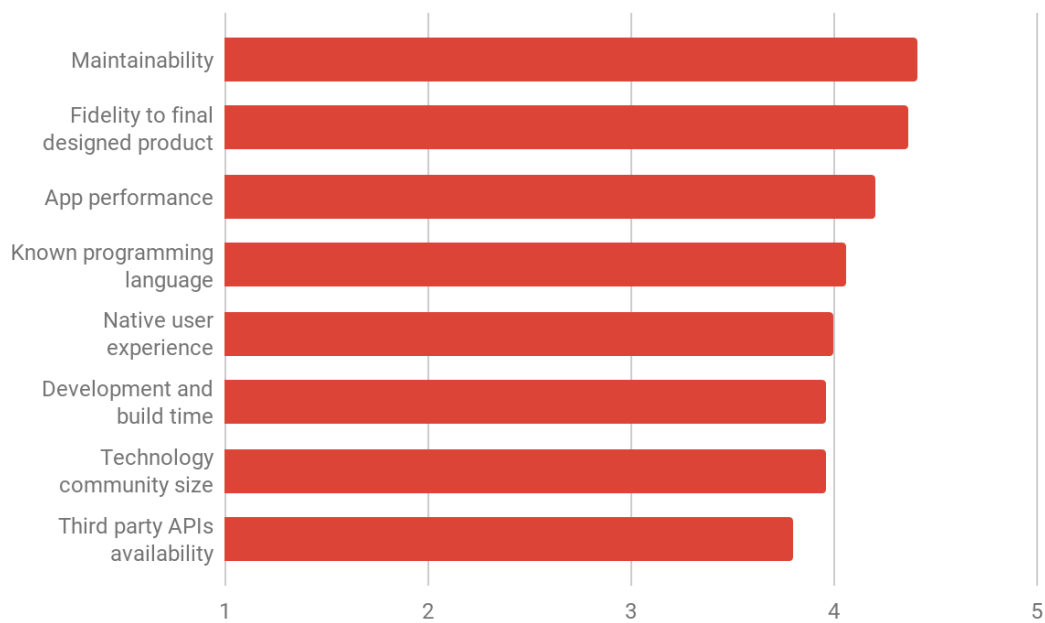


Figure 13. Bar chart that represents the scale regarding evolutionary prototype criterias [Own authorship].

Next, respondents said that ‘app performance’ is not necessarily important in the initial interactions of the prototype, but it should be a concern as soon as the initial prototype version is released. As for the programming language chosen, it was unanimous that it should reflect the project’s goals instead of the developer’s knowledge. It was also stated that it is important to know the limitations of the language and framework chosen from the beginning of the project to avoid future setbacks when the prototype evolves into a product. ‘Native user experience’ was considered important but it was pointed out that a native interface does not necessary means ergonomic, therefore, the layout itself was considered a priority.

The criterias between ‘reasonably important’ and ‘very important’ had a very similar score. ‘Develop and build time’ was the next ranked criteria. Respondents said that time should not be a priority when it comes to the evolutionary methodology since evolving a prototype means compromising to taking initial time to plan rather than building the application fast. It was also stated that the time is not as important

as the highest ranked criterias and, if necessary, time should be compromised instead of the app's layout and performance.

'Technology community size' was the next criteria. Even though it was ranked lower, respondents showed interest in a mature community when choosing a framework since it might save time when evolving the prototype. The last rated criteria was 'third party API availability'. Even though APIs might be beneficial regarding development time, it is not as important when compared to other criterias since the framework chosen is independent of APIs, therefore, not a meaningful factor in a long term methodology.

4.3 Discussion

In this subsection we aim to compare both prototyping criterias with one another, rather than discuss them individually. In order to visualize the results of this research, we chose to demonstrate them in **Table 1**, stating the importance of each criteria for each prototyping methodology. We will also compare each framework category based on the final outcome of the survey.

Table 1. Representation of final results and criteria evaluation [Own authorship].

#	Criteria	Description	Throw-away Relevance	Evolutionary Relevance
1	Programming language	The knowledge curve to acquire knowledge on this framework.	Moderately important	Very important
2	Community size	Maturity of the framework and resources available online to help.	Moderately important	Moderately important
3	Development time	Development time from creating the project to building.	Very important	Moderately important

4	Fidelity to final product	How easily it is to develop interfaces designed previously.	Moderately important	Very important
5	Native user experience	How close the application approaches a native experience.	Reasonably important	Moderately important
6	Maintainability	Resources to implement new features and fix bugs on production	Reasonably important	Very important
7	App performance	CPU usage, application loading time on user device, etc.	Reasonably important	Very important
8	APIs availability	Free or open source APIs available to reduce development time.	Reasonably important	Moderately important

A criteria that was not mentioned on the survey's options but showed to have value for throw-away prototypes was 'framework license type'. This criteria regards the monetary cost to develop an application that is similar to 'development time' that regards development time cost. Since company backed-up frameworks are usually more expensive and requires a licence, open-source frameworks are preferred by developers when building throw-away prototype.

Respondents also stated the difference among frameworks when building non-functional throw-away prototypes. iOS development showed itself to be a favorite among participants due to its ease of development and short time to build applications if a prototype requires only screen or UX validation. It was also stated that a non-functional framework could be used in order to save development time, since it requires very little skills to build interactive apps.

In general, respondents approved the use of a throw-away prototype when building features that require validation. That could be technical features that are part of a complex system or a small application that needs to be tested with its potential users. Either way, most of the respondents not only had used this prototyping

methodology before, they also strongly recommend it in the scenarios previously mentioned.

When comparing both prototyping methodologies, respondents attributed more importance to all the criterias in evolutionary prototype when compared to throw-away prototype. Since the code written in the prototype will be part of the final product, every one of the criterias must be carefully evaluated since they are requirements in market level applications.

A criterion that was not mentioned as survey option but demonstrated importance for evolutionary prototype was 'technology market life'. This refers to the time the framework has been on the market and was considered important since the prototype in this methodology aims to become final, therefore, a framework that has been on the market longer is consequently more mature.. This criteria showed itself to be similar to 'technology maturity size' that represents how active are the users of this framework.

Even though a big portion of the respondents had built an evolutionary prototype before, this methodology was also not recommended by a few of them when compared to throw-away prototype. Evolutionary prototypes might delay development time on features that need technical validation, such as complex or robust systems. Since features evolve with the entire application, wrong decisions can easily echo into the final product compromising time and money.

Part of the respondents stated otherwise, recommending evolutionary prototype as much as throw-away. If the time available to develop the entire application from scratch is short, the evolutionary prototype is a great approach. According to respondents, when this methodology is used correctly it can optimize development time if using a framework that develops and deploys for all OS such as hybrid or WebApp development.

Chapter 5

Conclusion

In this paper, we had the opportunity to apply a theoretical methodology into a survey that was sent and answers by dozens of developers across Brazil. With this research, we had a more practical grasp into what criterias modern software engineer consider when choosing the appropriate framework. It was interesting to observe the diverse range of expertise respondents had, raising very distinct answers and anecdotes.

The importance of this subject is clear since prototypes, such as MVPs and proofs of concepts, are becoming an essential step when building software as a whole. This happens due to the popularization of agile methodologies and design process when building applications, therefore, prototyping methodologies had shown themselves to be in an exponential growth.

With the dissemination of internet forums and development blogs, the information is becoming more diffuse and spread out. This makes it difficult for junior software engineers to find information such as the one approached in this work. We can also see the importance of researchers such as this one, in order to bring expert knowledge to novice developers in order to leverage the market regarding mobile technologies.

In our work, we made it easier for developers to seek the best approach for their prototypes based on the acquired answers on this survey. We did not conclude which frameworks are better for each scenario since it depends on a series of factors such as criterias priority, development team expertise, concepts that need to be proven, etc. Even thought, putting together these criterias as well as their importance when compared to each other will have a very positive impact in the next phase of this work, in order to build a complete guideline regarding mobile development and prototyping methodologies.

3.1 Future Work

This work has potential to scale in many different ways. Initially it would be interesting to launch a second survey following this one with mobile framework categories. In this next phase, developers would cite which frameworks are better for each prototyping methodology, through the match with the defined criterias. This way, we would have concrete answers in which frameworks work best for either throw-away or evolutionary prototype.

In order to grasp a better understanding of a worldwide context, it would be necessary to distribute this survey in other regions of the globe. Stackoverflow and Github are two platforms broadly used by developers that could be used to circulate the survey in english to other countries, as well as in different languages to have a wider range of answers.

It would also be possible to launch this prototyping research to other development platforms. Since prototyping is used in any scenario a system must be built from scratch, we could apply the concepts seen on this survey to other areas. In order to shorten the scope, mobile development was used in our paper, but it would be interesting to see how developers find these criterias adjust web development, desktop development and others.

References

- [1] HEITKÖTTER, Henning; HANSCHKE, Sebastian; MAJCHRZAK, Tim A. **Evaluating cross-platform development approaches for mobile applications.** In: International Conference on Web Information Systems and Technologies. Springer, Berlin, Heidelberg, 2012. p. 120-138.
- [2] WASSERMAN, Tony. **Software engineering issues for mobile application development.** FoSER 2010, 2010.
- [3] App Store Metrics. **148Apps**, 2019. Available on: <<http://148apps.biz/app-store-metrics/>>. Accessed on: April 4, 2019:
- [4] Android Market Stats. **AppBrain**, 2019. Available on: <<http://www.appbrain.com/stats/>>. Accessed on: April 4, 2019.
- [5] App economy to grow to \$6.3 trillion in 2021, user base to nearly double to 6.3 billion. **Techcrunch**, 2017. Available on: <<https://techcrunch.com/2017/06/27/app-economy-to-grow-to-6-3-trillion-in-2021-user-base-to-nearly-double-to-6-3-billion/>>. Accessed on: April 4, 2019
- [6] E. Masi, G. Cantone, M. Mastrofini, G. Calavaro, and P. Subiaco, **Mobile apps development: A framework for technology decision making**, in Proceedings of International Conference on Mobile Computing, Applications, and Services., ser. MobiCASE'4, 2012, pp. 64–79.
- [7] JOORABCHI, Mona Erfani; MESBAH, Ali; KRUCHTEN, Philippe. **Real challenges in mobile app development.** In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, 2013. p. 15-24.
- [8] CAREY, T. T.; MASON, R. E. A. **Information system prototyping: techniques, tools, and methodologies.** INFOR: Information Systems and Operational Research, v. 21, n. 3, p. 177-191, 1983.
- [9] KORDON, Fabrice et al. **An introduction to rapid system prototyping.** IEEE Transactions on Software Engineering, v. 28, n. 9, p. 817-821, 2002.
- [10] TATE, G. **Prototyping: helping to build the right software.** Information and Software Technology, v. 32, n. 4, p. 237-244, 1990.
- [11] GORDON, V. Scott; BIEMAN, James M. **Rapid prototyping: lessons learned.** IEEE software, v. 12, n. 1, p. 85-95, 1995.

- [12] LI, Hongwei et al. **What help do developers seek, when and how?**. In: 2013 20th Working Conference on Reverse Engineering (WCRE). IEEE, 2013. p. 142-151.
- [13] TENOPIR, Carol; KING, Donald W. **Communication patterns of engineers**. John Wiley & Sons, 2004.
- [14] HOLMES, Reid et al. **The end-to-end use of source code examples: An exploratory study**. In: 2009 IEEE International Conference on Software Maintenance. IEEE, 2009. p. 555-558.
- [15] U.S. Dept. of Veterans Affairs. **Veterans Affairs**, 2019. Available on: <<http://www.va.gov/trm/TRMGlossaryPage.asp>>. Accessed on: April 5, 2019.
- [16] Your Smartphone vs your PC. **AVI**, 2017. Available on: <<https://www.avi-b.com/your-smartphone-vs-your-pc/>>. Accessed on: April 24, 2019.
- [17] Native, web or hybrid mobile-app development. **IBM Software**, Thought Leadership White Paper, 2012. Available on: <<http://www.computerworld.com.au/whitepaper/371126/native-web-or-hybrid-mobile-app-development/download/>>. Accessed on April 24, 2019.
- [18] GORDON, V. Scott; BIEMAN, James M. **Reported effects of rapid prototyping on industrial software quality**. Software Quality Journal, v. 2, n. 2, p. 93-108, 1993.
- [19] DUC, Anh Nguyen; ABRAHAMSSON, Pekka. **Minimum viable product or multiple facet product? The Role of MVP in software startups**. In: International Conference on Agile Software Development. Springer, Cham, 2016. p. 118-130.
- [20] BRISCOE, Gerard. **Digital innovation: The hackathon phenomenon**. 2014.
- [21] BEYNON-DAVIES, Paul; TUDHOPE, Douglas; MACKAY, Hugh. **Information systems prototyping in practice**. Journal of Information Technology, v. 14, n. 1, p. 107-120, 1999.
- [22] PALMIERI, Manuel; SINGH, Inderjeet; CICCHETTI, Antonio. **Comparison of cross-platform mobile development tools**. In: 2012 16th International Conference on Intelligence in Next Generation Networks. IEEE, 2012. p. 179-186.
- [23] JOBE, William. **Native Apps vs. Mobile Web Apps**. International Journal of Interactive Mobile Technologies, v. 7, n. 4, 2013.

- [24] M. Kasunic. **Designing an effective survey**. Technical report, DTIC Document, 2005.
- [25] MOLLÉRI, Jefferson Seide; PETERSEN, Kai; MENDES, Emilia. **Survey guidelines in software engineering: An annotated review**. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, 2016. p. 58.
- [26] O. M. G. Omg, R. Parida, and S. Mahapatra. **Business Process Model and Notation (BPMN) Version 2.0**. Technical Report January, 2011.
- [27] TANG, Stephen et al. **Towards a domain specific modelling language for serious game design**. In: 6th International Game Design and Technology Workshop, Liverpool, UK. 2008.
- [28] Average Senior Software Engineer Salary. **Payscale**, 2019. Available on: <https://www.payscale.com/research/US/Job=Senior_Software_Engineer/Salary> Accessed on: April 26, 2019.
- [29] LIKERT, Rensis. **A technique for the measurement of attitudes**. Archives of psychology, 1932.
- [30] ALBAUM, Gerald. **The Likert scale revisited**. Market Research Society. Journal., v. 39, n. 2, p. 1-21, 1997.
- [31] About LinkedIn. **LinkedIn**, 2019. Available on: <<https://about.linkedin.com/>>. Accessed on: May 3, 2019.
- [32] Google Data Studio. **Google**, 2019. Available on: <<https://datastudio.google.com/u/0/>>. Accessed on: May 3, 2019.

A. Survey Questions

The data collection tool for this research was a public survey that can be found in this link: <https://forms.gle/zED3uH8ePwwZR24T7>. The first section of the survey contains questions regarding the demographics of the respondents and the second section there is the qualitative and quantitative research divided by prototyping methodology and each evaluation criteria defined previously.

Semi-structured survey questions:

1. Where do you live? (City/State/Country)
2. What's your gender?
 - a. Male
 - b. Female
 - c. Others
3. How many years of software development experience do you have?
 - a. Less than 1 year
 - b. 2 years
 - c. 3 years
 - d. 4 years
 - e. 5 years or more
4. And how many years of mobile development experience do you have?
 - a. Less than 1 year
 - b. 2 years
 - c. 3 years
 - d. 4 years
 - e. 5 years or more
5. What's your current role and company?
6. Define your knowledge on each of these mobile development categories:
 - a. Native (Android or iOS).
 - b. Hybrid (Ionic, Xamarin, React Native, etc).
 - c. WebApp (Angular, React, Vue).

7. Have you ever prototyped a mobile application before?
 - a. Yes
 - b. No
8. If yes, can you briefly describe its main features?
9. What frameworks, APIs and technologies were used?
10. In a scenario in which you have to build a throwaway prototype and the code will not be used afterwards, how relevant do you consider each of these criterias when choosing a framework?
 - a. Required skills to develop applications due to a known programming languages.
 - b. Technology community size.
 - c. Development and build time for each OS.
 - d. Fidelity to final designed product.
 - e. Native user experience for mobile users.
 - f. Maintainability and the cost of implementing new features.
 - g. App performance such as CPU usage and loading time.
 - h. Availability of third party APIs.
11. Can you elaborate about your choices above?
12. In a scenario in which you have to build an evolutionary prototype, maintaining its architecture and infrastructure so the code can be used afterwards, how relevant do you consider each of these criterias when choosing a framework?
 - a. Required skills to develop applications due to a known programming languages.
 - b. Technology community size.
 - c. Development and build time for each OS.
 - d. Fidelity to final designed product.
 - e. Native user experience for mobile users.
 - f. Maintainability and the cost of implementing new features.
 - g. App performance such as CPU usage and loading time.
 - h. Availability of third party APIs.

13. Can you elaborate about your choices above?

14. Do you think any criteria was missing? If yes, which ones and why?