Don't Panic

MOBILE DEVELOPER'S GUIDE TO THE GALAXY
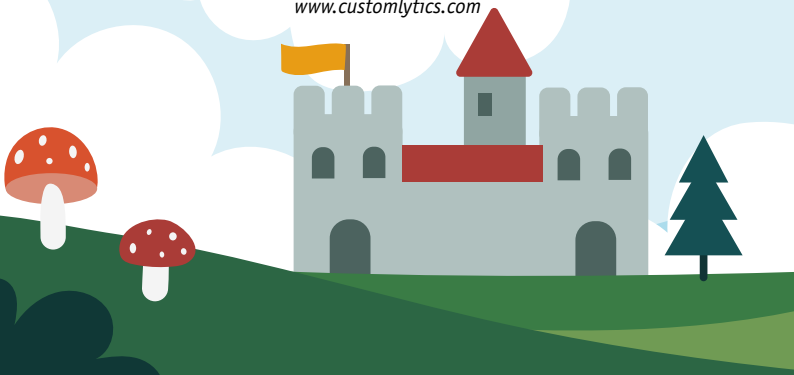
18th Edition November 2019

This Developer Guide is licensed under the
Creative Commons BY 2.5 License.

Please send your feedback,
questions or sponsorship requests to:
mobiledevguide@open-xchange.com
Follow us on Twitter: @MobileDevGuide

Art Direction and Design by

Cornelius Haverland
Sarah Duwe

Editors:

Marco Tabor
Mladenka Vrdoljak

www.mobiledevelopersguide.com

Mobile Developer's Guide
Contents

Prologue

The edition you are holding in your hands is the very first one that has been co-published together with Customlytics. We at Open-Xchange are happy to have found a partner that shares our goals and that allows us to continue printing hardcopies to provide our little book to an even larger audience. Most importantly, this partnership has a hugely positive impact on the content of the Mobile Developers Guide. Readers of previous editions will notice that the chapters about user analytics and monetization have seen massive updates and that we added new chapters covering topics like app store optimization, user acquisition and retention. In our opinion, covering these topics in depth was overdue. As the industry grows and changes, it has become clear that developing an app is just one part of the adventure. No matter how good your piece of software might be, you will not succeed without at least some knowledge about how to market it. We hope our guide helps you to learn and build upon this knowledge.

Mobile marketing has become highly data-driven these days. There are numerous 3rd party tools that allow you to, for example, attribute installs to their respective source properly and enable marketers to capture a huge range of user data at particular stages in their marketing campaigns. Before we proceed, we want to issue a word of caution to our readers when it comes to handling this (user) data. We advocate for a responsible way of collecting, storing and handling user data, which respects users´ right for privacy and which is in-line with the latest legal requirements such as GDPR. Throughout the book, especially in the marketing chapters, you will notice the reference to the use of these tools from time to time.

We make the case that using SDKs of independent 3rd party tool providers - in contrast to certain other "data hunters"- will give app developers sovereignty and control over their data. You can read more on this in the mobile analytics chapter.

Of course, you will also still find tons of updated information about how to create mobile software that is worth the marketing effort. So make sure to read all the chapters and understand how to convert your idea into code. Once you are finished reading this edition of the Mobile Developers Guide, we would like to hear your feedback. Please let us know what content you think we may have missed or where you see room for improvement. Or, even better, get involved and share your knowledge by becoming a contributor for the next edition.
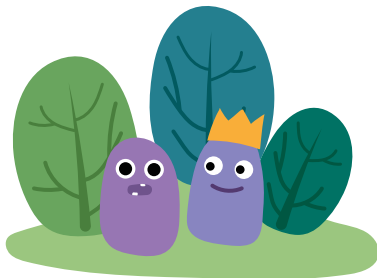
Cheers,

Marco / Open-Xchange & Christian / Customlytics

Bremen & Berlin November 2019

PS: Please follow us on Twitter *@MobileDevGuide* and visit *mobiledevelopersguide.com* to obtain the electronic edition of this booklet.

PPS: As usual, the articles reflect the opinions of the respective authors.

Robert Virkus

BY

The Galaxy of Mobile

Looking back into the past, the mobile scene looked a bit like the home computer ecosystem in the 1980s: many players, many operating systems, lots of innovation and changes. Today we are dealing with a global duopoly: 98% of today's mobile devices are running either on Android or iOS[1].

Will this change? Without political pressure or a paradigm shift we do not expect the situation to change for the foreseeable future. Looking at the details, however, reveals an exciting micro-cosmos of options, challenges & opportunities.

Notable Players

You should watch the following players carefully, obviously depending on your market strategy and app/game/service category. Of course, each of these players bring way more to the table than just the mentioned products and services, but these are the most relevant in the mobile segment that we cover in this guide. Some of these players form global digital leaders summarized by the GAFAM acronym, which stands for Google-Apple-Facebook-Amazon-Microsoft.

Google

In the mobile industry Google's Android operating system remains the major smartphone player - both by numbers and by revenue.

Android has many variants like Android Go for lower end smartphones, Android One promising regular security updates

---

1   76% Android, 22% iOS, see gs.statcounter.com/os-market-share/mobile/
     worldwide/2019

for three years and OS updates for two years[2], Wear OS for smartwatches and Android TV for smart TVs, and an integration option for cars called Android Auto. Then there is of course AOSP - the Android Open Source Project[3]-, which provides the core Android OS platform without the proprietary Google services. AOSP is used mostly in China but also forms the basis for LineageOS.

Additionally, Google taps into the emerging game streaming market with Stadia and has established a solid share of the smart speaker market with Google Home and Google Assistant-equipped devices from other vendors. With Chrome OS Google also established itself as an important PC operating system vendor. After Amazon and Microsoft, Google is the third biggest cloud service provider.

Last but not least, Google launched with Flutter[4] a cross-platform framework that spans across Android, iOS, Windows, macOS and the web. Flutter also powers UI development on its future research microkernel-based OS called Fuchsia[5].

Apple

Apple single-handedly created the smartphone market in the first place with the iPhone and specifically with the iOS operating system and the App Store in 2007 and 2008 respectively. Apple is also a leading tablet provider[6] and leads

---

2   android.com/intl/en/one

3   source.android.com

4   flutter.dev

5   fuchsia.dev

6   As of July 2019, iPads held 70% of the global tablet market: gs.statcounter. com/os-market-share/tablet/worldwide/2019

the smartwatch market with its Apple Watch product[7]. Last but not least they also have a solid multimedia offering based on the Apple TV with its tvOS.

Compared with Android, Apple has a comparatively smaller but very lucrative market share.

Samsung

Being the biggest smartphone vendor, Samsung influences the market to a large degree. Samsung is running Android mostly, but they also work on the Tizen operating system - however, so far every attempt to establish this OS in the smartphone segment has failed. With DeX (Desktop eXperience)[8] Samsung fuses the tablet and PC market by supporting keyboard, mouse and external monitors to work with selected hardware like the Samsung Galaxy Note 10 handset.

Samsung also operates the Galaxy Store for app distribution which can be an attractive alternative to the Google Play Store.

---

[7]   According to Strategy Analytics, Apple's smartwatch market share in Q2 2019 was 46%: bit.ly/2MkTFzK

[8]   samsung.com/global/galaxy/apps/samsung-dex

Huawei

Huawei has grown to become the second largest smartphone vendor. Same as for Samsung Huawei is using the official Google-Android for the international market.

Possibly the US administration triggered a new wave of innovation by forcing Huawei to move away from the Google-approved Android OS. Huawei presented its own operating system called Harmony to the public in August 2019. This is aimed to run on a Microkernel but the initial release relies on a traditional Linux kernel.

With App Gallery, Huawei also operates its own app distribution platform, which can be an attractive alternative to Google Play.

Facebook

Facebook provides the leading mobile messaging systems with WhatsApp, Facebook Messenger and Instagram. FB Messenger specifically provides easy extension options for developers while Instagram recently introduced an AR extension called Spark AR Studio. Facebook is also the main player of the Libra cryptocurrency project and provides various other developer tools. Rightfully Facebook is under pressure for its various data and privacy breaches.

Amazon

Amazon is the leading smart speaker provider with Alexa, which is extensible using skills. Amazon also uses its own Android variant FireOS[9] for its tablets and provides its own Android app distribution platform Amazon Appstore. Amazon also provides the leading cloud platform with AWS.

---

[9] developer.amazon.com/de/android-fireos

Microsoft

Microsoft is still the leading desktop operating system provider but failed to get traction with its Windows Phone and subsequent mobile offerings. Still, the desktop and gaming market make Windows and Xbox attractive targets for developers. Microsoft also cooperates with Sony on the emerging game streaming market using its own Azure cloud platform.

TenCent

With WeChat and QQ Chat Tencent provides the third and fourth biggest mobile messengers worldwide. With Mini Programs, WeChat provides an important option to discover services and games. WeChat Pay is one of China's leading mobile payment systems.

KaiOS

KaiOS[10] is an operating system aimed at low cost feature phones and is based on the Firefox OS. It has gained substantial market share in some regions specifically emerging markets[11]. With KaiStore you can distribute your HTML5 apps easily.

Sailfish OS

Sailfish[12] is around for quite some time and currently in its third generation. Traction so far is limited, but thanks to an Android compatibility it can run at least some Android AOSP apps.

---

[10] kaiostech.com

[11] In India, KaiOS has a market share of >4% which makes it the second most popular mobile operating system (gs.statcounter.com/os-market-share/mobile/india).

[12] sailfishos.org

Stand Out in a Crowded Market

With the increasing competition in the apps space there are various aspects that are worth considering:

— Experiences can carry across a variety of form factors - may it be in-car-systems, TVs, PCs, game consoles, augmented reality or voice-enabled smart home systems. As mobile technology moved to many systems, you can use your existing app development skills to reach these form factors. But make sure to adapt to each platform in the best possible way, do not limit yourself to the least common denominator!

— Users seem to be less likely to try and install new apps, therefore existing apps will increase their features - moving from a single-purpose to a multi-purpose app world.

— With multi-purpose apps, extensions play an important role. Instead of creating and maintaining your own app you can also extend existing apps such as WeChat, Facebook Messenger, Google Assistant, Apple Siri, Microsoft Office, or system extensions like the iOS FileProvider or the Today app extensions. Search for "zero UI" and "atomized apps" to learn more.

— Think about creating extension points in your own app as well, in order to allow others to get their services into your app.

— Take notifications seriously and make sure to add interaction options to your notifications.

— Driving engagement is - as always - critical. One of the biggest driver for apps is communication and socializing - also compare our brand new "Mobile Customer Relationship Management" chapter for learning more.

The focus of this book is on developing mobile apps, which encompasses a number of phases including: planning and specification, prototyping and design, implementation, internal testing and deployment, deployment to an app store, discovery by users, installation, use and feedback. Ultimately, we want our users to enjoy using our apps and to give us positive ratings to encourage other users to do likewise.

While developing, mobile apps share many common features with developing other software, it has specific characteristics. We will cover some of these next.

How to Service Mobile Devices

There are several ways to realize a mobile service:

Native Apps

A native app is programmed in a platform-specific language with platform-specific APIs. It is typically purchased, downloaded and upgraded through the platform-specific central app store. Native apps usually offer the best performance, the deepest integration and the best overall user experience compared to other options. However, native development is often also the most complex development option. When starting new apps you should consider using Kotlin for Android and Swift for iOS, rather than Java and Objective-C. Find further information on how to get started in the dedicated Android and iOS chapters.

Websites & Web Apps

Websites or - increasingly - so called single-page applications are written in a variety of languages and use HTML and CSS for rendering. Consider using progressive web apps[13] and configure them for iOS desktop pinning[14].

Web apps run without an app store, so you are independent of app stores which is both good, because you are not limited by the app store and bad, because it is harder for your users to find you.

Of course, you can also find a dedicated chapter on mobile web development in this book.

Cross-Platform Apps

There is a multitude of cross-platform services, that provide write-once run-everywhere scenarios. Even when dealing with only two dominant platforms, cross-platform tools can help you to update and maintain your services with less effort. Read the cross-platform chapter to understand your options in this regard.

Flamewars

As developers, we tend to have a passion for our chosen darlings. However, let us not forget that these technologies are just that - technologies that are relevant at a given time and in a given space, but not more. Yes, flamewars are fun but in retrospect, they are always silly. Hands up those who fought about Atari versus Amiga back in the good ol' 80s! Probably

---

13 developers.google.com/web/progressive-web-apps

14 developer.apple.com/library/content/documentation/AppleApplications/
    Reference/SafariWebContent/ConfiguringWebApplications/
    ConfiguringWebApplications.html

not many of you but, surely, you get the point. Initiatives such as FairPhone[15], ShiftPhone[16] or the GuardianProject[17] may prove more important than the OS or vendor of your choice in the future.

If you are lost in the vast space of mobile development, do not worry, stay calm and keep on reading. Go through the options and take the problem that you want to solve, your target audience and your know-how into account. Put a lot of effort into designing the experience of your service, concentrate on the problem at hand and keep it simple. It is better to do one thing well, rather than doing 'everything' only so-so. Invest in the design and usability of your solution. Last but not least, finding the right niche is often better than trying to copy something that is already successful. This guide will help you make an informed decision!

15 fairphone.com
16 shiftphones.com
17 guardianproject.info

BY Andrei Balaz

From Idea to Prototype

What makes a service successful? Why do so many start-ups fail, often despite their technological superiority? Why do so many products fall into the build trap[1] and end up stuffed with features that are used little, yet had cost considerable development and maintenance time?

One of the biggest pitfalls in product development is failing to understand how your product fits into people's lives. Not knowing what progress your customers are trying to make when choosing your product over another, can lead your team to build the wrong or over-engineer the correctly scoped product.

This might seem paradoxical at first. It would be completely reasonable to expect that companies would not invest into features that will rarely be utilized. Unfortunately, start-ups and established companies alike, often lack a process to analyze the reasons why customers hire their product in the first place. Instead, they frequently take the shortcut and decide what to build next based on some sort of "best-practice-remixing" of the competitors' successful features or uninformed ideation.

Building and failing fast is misinterpreted as a goal in itself. Teams forget that recovery from failure can be difficult and that pivoting needs to be done carefully and informed by the right type of data (more on that later).

It is in every team's interest to explore methods that allow them to uncover the most about their customers' contexts, motivations and struggles with as little investment as possible before they start building.

---

1   youtu.be/DmJXpI7OJuY

In this chapter we will explore:

— Why people buy new products
— What data is useful to drive innovation
— How to conduct pragmatic customer research to frame your product's value proposition and discover what to build and what not to build
— How to formulate insights and align your team around the progress that users are trying to make.
— How to prototype and test your ideas by involving your customers at every step

At the end of the chapter you will be equipped with a powerful perspective and a set of tools that will help you on your next endeavor to start building a service that customers will value.

Why do people buy products?

People do not buy products, they buy better versions of themselves.

In order to explain what this means, consider this example:

Remember the last time you purchased wine to serve it to your friends. How much did you know and care about the wine's properties and features such as vintage, grape, price etc.? If you are not a wine expert, I am sure the context of the upcoming meeting with your friends played a more important role in your decision than the individual features of the wine. The price and grape might have come up in your mind but the region and vintage probably did not influence your decision much. You might have simply selected a dry wine because you know that is what adults drink. Depending on the strength of

your desire to impress your friends, you might have selected wine in a wine store to make sure you are serving something respectable to your friends. Or you might have visited the supermarket to simply get more of it—you know, to get the conversation rolling faster.

No matter what the case, you selected the wine based on your desire to be a better host in that given situation. To speak in a software analogy, you wanted to upgrade the "version" of the host you were before. The attributes of the wine only played a role in your decision-making if you were able to connect them to this purpose in your mind. The wine helped you to "progress" – you hired it for the job of making you a better host.

Accordingly, when you interview a customer about a product, do not only talk about the product itself, focus on what they do with it. They hire the product to make progress in their lives, not only because of its features or attributes. This motivation to make progress arises when they encounter a situation where their current state is not good enough. We can describe this condition and desire for progress as a Job to be Done[2]. The 'job' stands for the desired progress a person wants to make. Since it is still only a desire and not reality yet (there are some constraints blocking the person), this job is 'to be done'.

Progress is not only functional.

It is important to note that when describing customer progress and the jobs they are hiring products for, it is not sufficient to think and talk only in functional terms (What they can do with a product). The wine from our previous example certainly needed to meet a certain standard to help you stand

---

out as a good host, yet being a better host incorporated emotional and social goals, as well. You tried to impress your friends and feel appreciated. Because most human decision-making and motivation is ultimately emotional, framing how your product helps your customers to improve always includes an emotional dimension (How I want to feel and how others should think of me).

Product do not have jobs, people do.

To truly understand what a 'Job to be Done' means, one must grasp that people and not products have jobs. One should not say that the job of a car is to move people from A to B. Instead, the owner of a car may have the job of getting to work on time or spending a great vacation on a road trip. The car is the solution hired for the job, and could have been substituted by many others (e.g. carpooling or hitchhiking) to help the person make the same kind of progress. The same product can be hired by people for wildly different jobs.

Jobs are not tasks. People do not want to do tasks.

One common misconception is that people hire products to do the certain tasks, such as cutting their lawn, or driving. Given what we stated before, this innovation perspective is misleading. It can bias teams to incrementally improve lawn mowers and cars and forget about solutions that make these tasks completely unnecessary.

People are interested in making progress in their lives, not in doing the tasks that are currently necessary to make that progress. If they find a solution that allows them to forget about mowing their lawn and still have a presentable garden, they will be very much inclined to leave their lawn mower behind and switch[3].

To get this point across, let us look closer at the 'Job to be Done' metaphor. People 'hire' products in the same way a manager would hire an employee to get a job done. The manager wants the benefits of somebody else's work, without having to do the work themselves. After all, that is why they hire somebody to do the job. People hire products in the same way. They want to enjoy the benefits and get their job done without actually doing the work. People do not want to do stuff, they want the benefits of that stuff being done for them. Ergo, the best interface is no interface.

Customers do not see competition in the same way many organizations do.

When customers experience that something is not good enough and start looking for something better, they consider a variety of solutions that might deliver the desired progress. The term solutions may refer to products and services, but it also includes behaviors, workarounds, a combination of the latter—sometimes even—doing nothing. Following on our wine example, a party host who is interested in entertaining his guests might consider hiring wine, telling friends to bring their own favorite drinks or not hiring drinks at all and hiring a DJ instead.

Notice how the customers consider solutions outside of a particular product category (e.g. alcoholic beverages) and

---

3   For example, they could use gene-modified grass or hire the neighbor to do the work.

how this perspective differs from a traditional way to look at competition. When challenged with innovation, companies often focus on their closest competitors (beer, soft drinks, other wine producers) and end up remixing the competitors' best features into one-size-fits-all solutions. Failing to acknowledge that this approach often leads to over-engineering good enough solutions, companies become blind to disruptive efforts from other markets.

By embracing the view that customers are hiring solutions to progress and get a job done and not because of their features and functionality, you enable your team to innovate more freely and learn to categorize the market in the way your customers do. As a wine producer, you might incorporate your primary product (wine) into a service ecosystem that helps your customers to become knowledgeable and fun party hosts.

Why is understanding people's desires for progress important for innovation?

Companies usually focus their research efforts on understanding the quality of their users' experience. They might collect usage analytics across different steps of the user journey, analyze their sales funnel, collect and sort frequent feature requests etc. While this data is immensely helpful and should definitely be gathered to improve the quality of the user's experience[4], one must understand that this is not the only data a company should collect, especially when challenged with new product development, scope creep challenges and growth.

The problem stems from the sampling bias[5] inherent to collecting data that originates only from analyzing the current product's usage. When only talking to people who already use the current version of the product, one can easily forget about people who are not customers yet and who might have left the product (churners). This practice leads the team to focus on improving of what already is (optimize and innovate incrementally), rather than thinking of something radically different (think gene-modified seeds as opposed to an improved lawn mower). This can inhibit growth and lead to bloated or highly specialized products for a small group of vocal users. Collecting only so-called 'data for quality improvement' does not allow teams to think beyond what already is and reimagine products radically, opening up space for unbiased competition and disruption.

The Jobs to be Done perspective provides a helpful way for analysis of underlying motivations, goals, constraints and other factors that create and influence demand for a new product

---

4   see the Analytics chapter to learn more about this important aspect

5   en.wikipedia.org/wiki/Sampling_bias

before the product is used. It provides the 'why' behind the 'how' and the right mental model for new market innovation. We can call this additional type of data Jobs to be Done research provides, 'data for growth.' 'Growth data' can be used to explain analytics, inspire ideation of new solutions and inform market positioning.

Only when 'data for quality improvement' and 'data for growth' are both collected and understood, a team is equipped to sense when to optimize or to pivot and innovate.

Why do people buy a certain product?

We have established that it is desirable to concentrate on helping your customers to become better versions of themselves when building new products. But how do you find out how this better version looks like? How do you find out which job your customers are hiring your product for?

The most pragmatic way to uncover what motivates people to progress is to research their shopping and decision-making behavior in qualitative interviews. These so-called 'switch interviews' allow you to understand what progress they were seeking and what was important to them as they chose a product. To understand better how we can describe the jobs people might have, let us look at the following data model.
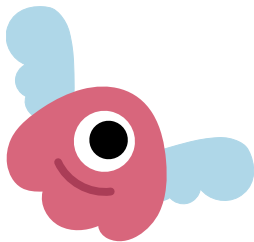
The Jobs to be Done data model

As people go about looking for a new solution, they want to reach 'unmet goals'. These goals are unmet because some 'constraints' exist that prevent people from reaching them. People become aware of their unmet goals and constraints when some 'catalysts' happen and trigger them. To meet their goals, they hire a set of solutions from which they choose one or a combination of a few. We call this last element a 'choice set'.

These four elements combined create demand for something new. They create the aforementioned 'Job to be Done' in people's lives. Understanding them helps us describe what people are looking for in a new product and design it accordingly. Let us look at each in detail.

Unmet Goals

Unmet goals describe experiences we want or want more of, but cannot have at the moment because of constraints. The goals can have various types, for example:

1. **Competence:** Become very capable and effective in your actions rather than feeling incompetent and ineffective.
2. **Control:** Determine the behavior of, or supervise the running of things around you.
3. **Relatedness:** Have regular intimate contact with people who care about you rather than feeling lonely and uncared of.

4. **Self-Actualizing:** Develop your best potentials and make life meaningful vs. feeling stagnant and without meaning.
5. **Security:** Have safety in your life, rather than feeling uncertain and threatened by your circumstance.
6. **Others** include Influence, Physical thriving, Self-esteem, Pleasure, Self-expression, Recognition, Care, Insight, Self-proficiency, Scale, Time-saver, Risk-reducer, Reliability.[6]

Constraints

Constraints prevent us from fulfilling our unmet goals and make progress. They create a discrepancy between where I want to be and where I am now. Constraints usually relate to unmet goals directly. The types of constraints can be:

1. **Anxiety:** Concern about undesirable things happening.
2. **Lack of skill:** Lacking of knowledge to understand something or lacking the physical capability to execute.
3. **Switching cost:** A perception that making a change is too much effort, or will not be worth it.
4. **Habit:** Mental schemas and physical behaviors that one is accustomed to, that are hard to unlearn or disengage from.
5. **Organizational Barrier:** Attributes of the current work environment that are outside of one's control
6. **Others** include Functional dependency, Time, Money, Situational Barrier, etc.

---

[6] Read more in 'The Jobs to be Done Data Model' by Alan Klement: jtbd.info/the-jobs-to-be-done-data-model-b270f6fc445

Catalysts

These are events which create or affect unmet goals,
constraints and the choice set. They make people aware of
something not working or trigger action. They can be:

1. **Anticipated Events:** Expecting that something will happen, for example a deadline.
2. **Unexpected Events:** Something that caught you by surprise, for example a thing breaking.
3. **Advertisement:** Promotional content intended to inform you of a product or persuade you to use or buy it, for example a discount.
4. **Word of Mouth:** First-hand information gained from someone you trust, for example a friend's recommendation.
5. **Lower-order Job to be Done:** Seeking progress somewhere else that causes a desire to progress elsewhere, for example mastering cooking of a recipe to shine at a dinner party.
6. **Others** include Repeated events, Observed use, Positive experience with a product, Negative experience with a product, Making progress, Job getting done, etc.

Choice Set

The choice set describes everything that people consider hiring
to make progress towards their unmet goals or overcoming
their constraints. These can be products and services, but also
behaviors or combinations of products and services. Imagine
you wanted to impress friends at a dinner. You might choose
a delicate wine, but you could also bring a surprise or a great
story.

Switch interviews

Talking to customers who have switched to your product is an effective way to uncover why your customers hired your product and surface their unmet goals, constraints, catalysts and choice set. Switch interviewing is all about getting the story of your customers' shopping and decision-making on record, understanding all the causes and moments that led to hiring the product you want to analyze – yours or your competitor's. In the following, we will look at how to conduct these interviews. Mastering this technique will help your team to frame the design problem for your service and spot valuable innovation opportunities.

You start your interview by asking questions about the moment the customer purchased your product (The Switch). During the interview you retrace the story from the switch to their first thought, when your customer thought that what she had was not good enough anymore.

You then try to understand their 'old me', what constraints they were trying to overcome and what catalysts played a role in triggering that demand for change. You ask about how they imagined their life to get better once they found a new solution, putting some light on their unmet goals and formulate their idea of a 'new me'.

To help you with this retrospective interviewing process, here are some questions you can ask:

Start with the Switch:

1. Can you tell us when you started thinking about buying or signing up for our service/software?
2. Can you remember why you decided to make a change?
3. Was buying this product related to a problem you were having? Or a new initiative?

4. Did you have to convince anyone else to buy our product?
5. You first started thinking about buying our product a few months before you actually bought it. Why did you decide to buy when you did? Why not a month before? Why not wait another few weeks?
6. What other products were you thinking about using instead of ours?

When appropriate, ask more questions to get all elements of the data model:

1. How did you think things would be better for you personally once you had the product in place?
2. How did you think things would be better for your company or your customers?
3. Did you have any problems convincing others to use our product?
4. What other things were happening within your company during this timeline? Were there any changes in leadership or strategic direction? Did any of these influence the decision to start using our product?
5. Once you started using the product, how did you know things were progressing as you had hoped (or not)?

When wrapping up the interview, make sure that you understood the unmet goals and constraints the person had as they chose the product. Reiterate with them if your understanding seems right to them. Do not be worried to highlight contradictions or things that are still not clear to you. Switch interviews are engaging conversations that focus on the person's personal journey, not the product. There are no right or wrong answers.

Last, but not least, make sure to compensate people for their time.

Debrief directly after the interview and create a profile for each interviewee. Use the data model and collect what you heard for each category. Here is a Demand Profile template to help you with that: *bit.ly/mdgg-demandprofile*.

Who to interview

It is important that you talk to people who have already bought and used your product who have a somewhat fresh memory of all the important tradeoffs and circumstances that shaped their decision. Because people are prone to rationalize their decisions, avoid talking to customers who are only thinking to buy but did not commit yet. Similarly, talking to people who got a product gifted will not reveal any valuable insights—they simply did not spend any energy thinking about it. You will use the story of the shopping and decision-making process as a tool to identify customers whose motivations have undergone the test of reality, getting a clearer view on their actions as opposed to their aspirations and beliefs.

As a guide: Interview customers who switched to your product in the last 90–120 days. They will not be in love with the brand-new product anymore and will have had time to evaluate if they were able to reach the desired progress or not.

What if my product does not have customers yet?

It does not matter whether your app (or app feature) is already on the market or not. While interviewing current customers is easier for beginners, what you are after is the story of change. Once customers leave their old way of doing things in order to progress (for example starting to pay for premium features), they can be interviewed. In case your software is not released yet, you can simply interview people who switched to a similar competitor's product. In case you cannot think of any competitor or your app is free, talk to customers who exhibit

a new behavior that is in line with the progress you are trying to enable.

To give you an example: To create a new service that helps people to become better hosts, you could interview people who hired wine, DJs, a cooking class or use a new recipe app and see what motivated them. Be creative and sensitive to what people do today to improve in a similar way or situation.

For further tips, please read this article on Medium: *jtbd. info/uncovering-the-jobs-that-customers-hire-products-and-services-to-do-834269006f50*.

Define the problem: Job Statement

After you have conducted 10–15 switch interviews, you should be able to spot patterns emerging in the data that shaped the customers' desire for progress and demand for a new product. The profiles you created after each interview will help you to analyze the data in order to identify commonalities in behavior and decision-making that people demonstrated. You can create a super-profile, or in other words, a 'demand segment' for each group of similar unmet goals and constraints combinations.

Furthermore, it is helpful to summarize these insights in a structured and concise manner. This process will streamline the communication with your team and will help you to see which parts of your product are essential and which you can abandon.

The so-called 'job statement' helps your team to establish a high level vision for your project. It expresses what your customer is struggling with and what job they desire to get done. It opens up space for ideation by capturing insights into the customer's current state and motivation to progress. And it helps to shape your product's value proposition and aligning your company's strategy around a customer-centric purpose.

A job statement has two parts: 1. The situation in which a

desire for progress arises ('old me') and the future. 2. Desired state once the job is done and the progress has been made ('new me').

For a grocery shopping application, the job statement might be: "Free me from the stress of looking for healthy ingredients (STRUGGLE), so I can rekindle the fun of cooking for my family (STRUGGLE IS RESOLVED)."

A job statement should not include any solutions or features, otherwise it will limit your team's creative potential. Consider this job statement that ignores this rule: "Deliver groceries to my home, so I spare time shopping."

The latter statement describes the functionality of a grocery shopping app well, but it does not provide room for broader thinking. Solutions, such as ready-made inspirational boxes sold at the supermarket or recipe collections would not be captured by it. Observe further, how the goal of the second statement is creatively limiting ("so I spare time shopping"). "Sparing time" does not describe a desired future state, a 'new me'. It is an effect that might alleviate a problem people experience that blocks them from having fun while cooking (shopping takes time and is stressful), but it does not describe how the life the customer would improve.

For further tips how to formulate your job statements, consult the free book "When Coffee and Kale Compete" by Alan Klement[7].

[7]   whencoffeeandkalecompete.com

Define Situations: Job stories

After having formulated demand segments and a high level vision of your product using the job statement, you can now use Job Stories to describe concrete struggling moments in your customers' lives. They capture all the different struggling moments, contexts, anxieties, goals and emotions that you heard throughout your interviews. While job statements and demand segments give you a vision and the larger picture of someone's motivation, you can use job stories to frame smaller design problems to kick off the ideation process for an individual product within the scope of a service or a particular feature.

Job stories explain how a particular group of customers acted in a particular, concrete situation, what they were doing, what effects a solution should have in their mind and in which direction they wanted to progress. Let us have a look at an example job story for a grocery shopping application:

When I feel bad about shopping ingredients for my children that are not organic because visiting multiple stores with my small children is very exhausting,

I want a way to avoid having to shop at multiple locations,

so that I can serve my family healthy meals without trading convenience for quality.

When... describes the situation in which a problem arises and the person is looking for a change. What happened?

I want to... captures the expectations that customers have regarding the effects of a solution when they use it. The solution itself or its features are not mentioned.

So I can... describes what happens when the struggle is solved. It describes how life will be when the solution is effective in getting the job done.

If you are familiar with user stories, you will recognize that

job stories are fairly similar in their structure. The important difference is that job stories attempt to tell a story of progress instead of describing the needs of a particular persona towards reaching a goal. The reason for this is the observation that very diverse people can act similarly when they find themselves in the same situation. Consider this example:

A wealthy business man has about 25 minutes to pass security at the airport. He is hungry and sees that the line at the checkpoint is pretty short. He figures that he can grab a Snickers quickly to recharge before entering the nasty procedure. He enters a kiosk, grabs the chocolate bar and pays. He knows that a Snickers is not the best choice and goes against the advice of his nutritionist, but he enjoys it anyway.

Now try to answer this question: Which elements of the story had stronger influence on his decision? That he was wealthy, a businessman and is concerned about his nutrition or that found himself in an airport with limited offerings and time and an appetite for a quick bite? People quickly override their stated preferences and attitudes when they are confronted with a situation or social context that makes alternative behavior more favorable. This is the main reason, why asking people what they would do or what they want is such a risky activity. People adapt. It is not who they are, it is where, when and with whom they are that causes their behavior.

However, many software development teams work with "personas", fictional characters representing certain segments of their target group.

But as we discussed before, it is not who the customers are in terms of their demographic and psychographic group, what they state they want, how they look, where they live or what brands their prefer which causes them to switch. While these data can correlate with certain behavior, they do not cause it. Context does. It is therefore to frame your customer insights around situations and the behaviors and tradeoffs they cause, not around personas.

That is why job stories offer an improved way to frame insights into customer motivation and can be used as a foundation in agile development processes. While going in-depth on how to turn these insights into a complete product roadmap would extend beyond the boundaries of this book, let us have a look at a few helpful tools that help your team to frame what you have learned and start building:

— If your team is used to working with personas, reduce the amount of meaningless data and focus on customer motivation[8].
— Intercom offers free ebooks[9] on topics such as product management, jobs-to-be-done and starting-up. They also provide a format called Intermission[10] that can help you kick off a project. It is a brief project that tells a story of a customer wanting to progress which contains a couple of job stories and ways to measure the team's progress towards solving the customer's problem. To see how this works in real-life, read their blog[11].

8   slideshare.net/AndrejBalaz/improving-personas-with-jobs-to-be-done
9   intercom.com/books
10 blog.intercom.com/accidentally-invented-job-stories
11 blog.intercom.com/how-we-build-software

— Eric Ries and his Lean Startup methodology[12] provide you with a great toolkit and perspective in order to avoid building features that customers will not use and ensure that you do not miss opportunities for improvement as you build. Ries recommends a scientific approach of formulating customer-centric hypotheses, building solutions and prototypes as soon as possible, testing them quickly and analyzing how customers react with the help of clear metrics.

Define the Use Case: Customer Journey

Before you dive into building complicated prototypes or drawing wireframes digitally, it is essential that you sit down with a couple of sheets of folded paper or a block of long, screen-sized post-its and attempt to sketch how your customer will use your app.

Create a storyboard that captures the user's physical surroundings, who they are with and what they are interacting with. If your service has more than one touch point, for example if it enables the customer to share a ride and both the riders and drivers have an app, you need to think about the interactions between these two people. Your goal is to get the story and different interactions right. In this way, you will quickly notice and improve inconsistencies and get a feeling of which portions of the customer's journey are most challenging.

Pay special attention to what happens outside of the screen's boundaries. In some circumstances, users cannot hold their phone in a hand, for example when moving around at airports with lots of baggage in tow. This will enable you to

---

12   theleanstartup.com/principles

open up your imagination to other forms of interaction. Think about a bus service. It combines physical, as well as digital touchpoints that together help reduce the customers' anxieties and guide them through the experience.

Create the User Interface: Prototypes and Visual Design

When you understand how individual journeys fit together in the bigger picture you can move into more defined design.

It does not matter whether you have a big budget or are working on a personal project over the weekend, having a fairly complete prototype of your app is the best way to communicate your concept and discuss it with others. Prototypes are usually developed before you spend time on implementing code and pixel perfect design. An agreed clickable walkthrough is a useful reference that teams can work towards without risking going too much off track. An interactive prototype is the best way to visualize and evaluate your app's interactions. It is usable enough to communicate the design, so you do not need to provide as much documentation as you would need to annotate static images. Your prototype can have visual design applied and look exactly as it will after the implementation, or you can stay on the wireframe or sketch level and focus on functionality and content. Even working with paper, prototyping might be a viable option at an early stage.

Google Ventures Design Sprint

The design sprint methodology by Google[13] helps you to develop a tested prototype within 5 days. If you follow Design Sprint 2.0 by AJ&Smart[14], you can run it in 4 days. It does require user research before you start, but if you followed our suggestions so far, you should be equipped to start it right away. In the beginning of the week you decide on scope and the experience you want to design, then you progress to ideation, sketching and building of a testable prototype. By the end of the week you will have tested your idea with real users, gathering invaluable feedback about your idea.

The Design Sprint will get you a good overview of various helpful methods that you can apply to prototyping and sketching. In the following section let us highlight a few methods that will help you to transform your idea into a more coherent concept.

Prototyping & Wireframing Tools

While paper should remain a truthful companion to your design process all along the way, teams around the world have spawned great prototyping tools that help you test your ideas directly on the device and share it with others. They allow you to further elaborate your ideas, test concepts quickly and work on your devices. Below you will find a few tools that work well together.

---

13   gv.com/sprint
14   youtube.com/watch?v=Z8MOwcqZuuU

Quick prototyping with Sketch and InVision+Craft or Marvel

Sketch is a design tool for Mac that has become the de-facto standard for designing user interfaces. Its community offers hundreds of useful plug-ins and Sketch keeps improving the tool at a rapid pace.

Sketch is easy to learn and integrates well with tools such as InVision or Marvel. The latter allows you to build lightweight clickable prototypes that can be previewed directly on your device and speed up your design process (especially thanks to the many helpful real-data import features from Craft by InVision).

The prototypes are great for communication with your client (if you have any) and early usability testing. We recommend to make your designs interactive early on to spot problems with the interaction flow. Viewing your designs on your device is the only way to take the right visual design decisions regarding type sizes, whitespace and many other usability considerations.

— **Sketch:** *sketchapp.com*
— **InVision:** *invisionapp.com*
— **Marvel:** *marvelapp.com*

High-fidelity prototyping with Framer, Principle

If you are looking into prototyping animated flows and transitions that feel like the real thing and are programmed, Framer will be your first choice. Framer is based on React, which is a complex web framework with a steep learning curve, but can therefore easily transfer prototypes into the real-world environment. If you prefer to postpone any coding, Principle provides you with a simple interface to make interactive prototypes reality. It has its limitations when your prototype requires many states, so be prepared to "fake" things a lot. Both tools are well-documented and integrate well with Sketch.

— **Framer:** *framer.com*
— **Principle:** *principleformac.com*

Keeping specs and assets in order with Zeplin and InVision Craft

When transferring your designs to development, exporting of specifications and UI assets have been a painstakingly time-consuming process. Zeplin, as well as InVision help in this regard, allowing you to view information about margins, color values, code snippets at one glance interactively. While InVision Craft eliminates the need to spec designs, Zeplin goes further by simplifying exporting of assets. Both tools allow you to organize assets in helpful libraries that greatly improve cross-team collaboration.

— **Zeplin:** *zeplin.io*
— **InVision:** *invisionapp.com/craft*

Windows: Affinity Designer, Adobe XD

If you are working with a Windows computer, your options are a bit more limited, but you are not left alone. Affinity Designer is an excellent program for graphic design which allows you to create user interfaces as well. It is fast, cheap and supports reusable components. If you are familiar with Adobe Illustrator and Photoshop, you should feel right at home.

If you have the Creative Cloud subscription, you might give Adobe XD a try. While still relatively young, it is faster than any other Adobe application in creating and exporting clickable UI prototypes while sporting an easy-to-use interface. It has made great strides in catching up to Sketch, but it lacks its community and plugin ecosystem.

— **Affinity:** *affinity.serif.com*
— **Adobe XD:** adobe.com/products/xd.html

Visual Design

Unless you are building an app that uses a non-visual interface, your app's UI will rely on graphical elements, animations, colors, illustrations and to the largest extent typography. You probably clarified conceptual aspects of user interface design in the prototyping phase. You decided on where to place interface elements and what copy to write so that the user can progress through the app with as little friction as possible. It is a good practice to also think about the size of visual elements and typography while prototyping and creating wireframes. Common mistakes include making copy too small and hardly legible, cluttering screens with too many elements and not thinking of animations and transitions. Tools such as InVision or Marvel help you to wireframe directly on the device and streamline your navigation flow and typography very early.

Besides the obvious usability benefits, a clear and well-designed interface provides, visual design is key to making your app memorable and unique. If your service includes multiple touchpoints, following the same design principles across all of them becomes increasingly important. A consistent and well-designed experience is a crucial part in how your customers perceive your brand. No amount of visual finish can make up for a cluttered interface or a buggy product. By adhering to the platform's design principles, providing a frictionless experience, staying focused in your visual presentation and providing unique functionality that helps people to progress in their lives, you will be able to transcend the common misconception that a brand experience can only be evoked through colors, logos and styleguides.

Visual consistency throughout the flow helps users make sense of your UI and learn interactions faster. Decide on a primary action color and limit the amount of accents you use for purely decorative reasons. Your user should always know which actions are most important and possible in each view.

Classic principles of graphic design[15] and the platform's design guidelines help you to guide the user's attention and navigate throughout the app. Your app can look very different depending on which platform it was designed for. Make sure that your design follows the recommended practices for font use, standard icons and layout conventions. See the platform-related chapters of this guide to find more information and links to specific resources.

Finally, the launch icon is the first impression visual element that your app will be identified by and judged on. Make it look clean and easy to recognize, even in small sizes. Do not include your company's logo unless you turn it into an icon.

---

15 learndesignprinciples.com

User Testing

The best way to validate your interface concept is to show it to users as soon as your work is representative enough to prompt feedback. You do not have to wait until you have a finished and polished product. Testing early can save you a lot of time in the long term. It will expose concepts that do not work early in the process. The more time you invest into developing your designs, the harder it gets to let go of them and start over. It is more difficult to accept feedback on something that you considered almost done, than on a clickable prototype that you can update quickly.

Test your assumptions regarding interaction, visual design and content as often as you can. It will help you to streamline your design process, uncover problems quickly and generate new ideas.

The best way to test your designs is to invite some users over and watch them perform your testing tasks. You will quickly see where they struggle and will be able to invite stakeholders to sit in or watch a live recording of the test to help your team members empathize with your customers. A typical user testing session is about an hour long. During that time users that are unfamiliar with the product are asked to perform certain tasks, usually around core functionality.

Here are some helpful tools that will greatly improve your process:

1. Build a testable prototype
2. Set up your hardware for recording. Use Reflector 3[16] and QuickTime or Lookback[17] to record your screens and cameras. Try to record the device and the user's face for later analysis.
3. Print a short introduction for your participant. Imagine a description similar in length and style to those in the App Store. (Value proposition, short)
4. Print every task on a separate sheet for your participants to read.
5. Recruit participants and invite them. When searching for people to interview it is good to refer to the original personas descriptions and look for users that match those profiles. Offering rewards makes it easier to find appropriate candidates, but be careful when choosing them: You do not want people to show up because they get money, you need people that fit your target group and give you honest feedback. So consider offering rewards that fit your target group, e.g. a photo printing voucher if you are testing a photo app.
6. Introduce yourself, clarify that there are no right or wrong answers and that the person should think aloud. Let them know that you want to hear everything they think when they do something with your prototype. Mention that the prototype is not complete and there might be some unfinished parts. If they assume that the person that is running the session is the author of the design, they might feel cautious of giving critical feedback. Reassure them that they are free to express their honest opinions.

---

16 airsquirrels.com/reflector
17 lookback.io

After all, the only reason you arranged the testing session was to get an independent feedback.

7. Let your participant read the introduction and then perform all tasks.

8. Do not lead users to any conclusions. Do not help them out by revealing how things work (unless they cannot figure it out and you cannot proceed with the session) and word your questions in a non-interruptive way.

9. Give a reward to your participant.

10. Discuss key insights directly after the test with your other team members. What went well, what went bad, what was surprising? As you progress through the tests with 4–8 participants, patterns will start to emerge.

For a more robust approach, use Lookback which allows your users to participate in your test by simply downloading an app to their device. Not only can you record what your users are doing, you can choose to perform in-house tests, let users perform the test on their own (unmoderated testing) or jump into a live session with them (moderated testing). The Lookback tools are thorough and easy to use and can spare you lot of time and fiddling with your own setup.

If you want to spare time with recruiting and recording your participants, you can use platforms such as *usertesting.com* to facilitate your testing process. You will receive recordings of your users as they are confronted with your prototypes and tasks. The biggest advantage of *usertesting.com* is their large pool of potential users, allowing you to recruit people from different countries and demographics. The recruiting criteria do not allow you to select people based on their situations, struggles or switching behavior. But on the plus side being able to let your tests run over the weekend is a great time-saver. The downside is that your prototypes need to be more

refined than in an in-person test where you can moderate if something breaks.

When you get feedback, you can reiterate your design and improve the parts that were not quite complete or if the feedback was good move on to the development phase.

If you are still exploring new areas and your own prototype is not quite ready, you can run testing sessions on other apps that have been already released. It can surprise you how much others notice about the application that you might never have thought of.

Keep Iterating and Learning

As you build your product, it is important that you test your assumptions frequently. Not only will you be able to uncover what works and what does not before committing large sums of money and time to building the wrong features, you will build empathy for your customers within your team.

Users, their motivations, struggles and circumstances change continuously, so it is crucial to develop processes within your company to gather feedback at various levels of the creation process.

Successful innovation does not have to be guesswork, but it will always remain a study of complexity. The sooner you develop your own process of learning and understanding, the sooner will you be able to qualify your customers' feedback, take better scoping decisions and find customers that will value your efforts. On to the challenge!

Daniel Böhrs

BY

Android Development

The Ecosystem

The Android platform is developed by the Open Handset Alliance led by Google and has been publicly available since 2007. Its use by the majority of hardware manufacturers has made it the fastest growing smartphone operating system ever which today dominates the market: More than 75% of the mobile operating system market share worldwide[1]. At the 2019 I/O developer conference, Google announced that there are 2.5 billion active Android devices[2] which also includes wearables, tablets, set-top boxes, TVs, phones and car entertainment systems.

The term Android describes a whole ecosystem beside the operating system itself. Google and multiple other companies like Amazon are providing stores, default apps and other components not directly bundled with the operating system itself. All required components combined, provide the user with an extendable system for daily usage. Every device licensed with Google is allowed to use the Google Play Service and the Google Play Store. The store provides around 2.4 million apps[3] and is the biggest store for mobile applications world wide. The proprietary Play Services provide access to Googles own APIs, like Google Maps and the Google Assistant.

---

1   gs.statcounter.com/os-market-share/mobile/worldwide

2   theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote

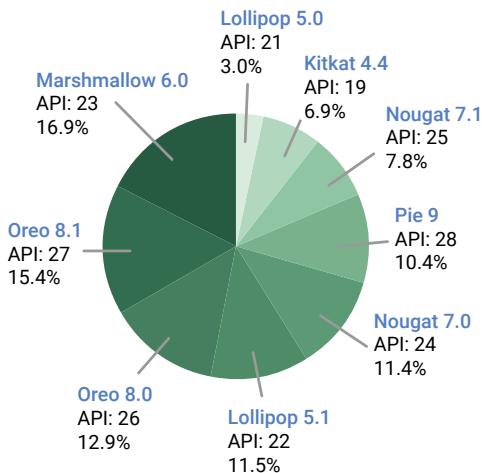3   statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores

Beside the big players like Google and Amazon, also so called custom ROMs, developed by the Android community or other companies can be used. Those often use alternative stores to provide apps. An example is the F-Droid[4] store, which only allows open source apps. Utilizing those, allows Android users to get a Google free experience if preferred. The Android ecosystem is a huge and flexible system for software on different hardware devices.

Due to a huge amount of manufactures, devices and adaptations of Android, one of the biggest problems of the platform is the fragmentation of the used versions. This leads to uncertainty over whether or not your Android application will run everywhere and often requires additional work by the developers. Furthermore, the adaption of the latest OS version is slower compared to other mobile platforms. However, in the last years Google increased efforts to reduce the fragmentation and today you will reach around 90% of the installation base if you decide to target 5.0 Lollipop or above[5].

---

4   f-droid.org
5   developer.android.com/about/dashboards

Lollipop 5.0
API: 21
3.0%

Kitkat 4.4
API: 19
6.9%

Nougat 7.1
API: 25
7.8%

Pie 9
API: 28
10.4%

Nougat 7.0
API: 24
11.4%

Lollipop 5.1
API: 22
11.5%

Oreo 8.0
API: 26
12.9%

Oreo 8.1
API: 27
15.4%

Marshmallow 6.0
API: 23
16.9%

Data collected during a 7-day period ending on May 7, 2019. Any versions with less than 0.1% distribution are not shown and devices without Google Apps (like those from Amazon and many China-based manufacturers) are not counted. All versions prior Android 4.4 were also excluded due to their outdated nature.

Recent Versions

Once a year Google releases a new Android version with features, improvements and security fixes. This also means new possibilities and challenges for developers. In preparation for updates, developers should also check for usage of deprecated or removed SDK functions or APIs, to avoid incompatibilities.

Android 9 (Pie)

Android 9 (codename: Pie) was released in August 2018. With the Adaptive Battery and Brightness features, Android 9 automatically adjusted apps, services and the brightness depending on the users habits, to further extend the battery time.

It also introduced App Actions[6]: Powered by machine learning, Android tries to predict user behavior and offers shortcuts based on detected individual usage patterns.

The interaction model for recent apps, which allowed users to quickly switch between apps was another new feature of Android Pie.

Just like iOS, Android also allows users to keep an eye on their screen time since this release: A dashboard was added to visualize the number of times apps where used. Users can also limit usage times for apps and enable automatic grayscale or Do Not Disturb mode.

As notch hardware designs are getting more and more traction Android 9 was the first Android version supporting it.

From a developer's point of view, the most relevant news in Android 9 included new indoor navigation features using indoor positioning with Wi-Fi RTT and enhanced notifications, e.g. the new Person class, which allows providing richer information.

It is now also possible to access cameras simultaneously and create for example streams with picture in picture content.

A new image decoder, a class for image animations, better codecs and media APIs aim to provide developers more options to create optimized multimedia applications. The Neural Networks API was also updated to provide more operations.

6   developers.google.com/actions/app

Android 10

With the most recent Android release in September 2019, the naming scheme has changed. Google has long named its Android software after sweets. These times are over. To avoid confusion, the latest version is simply called Android 10 without any additional codename.

Android 10 brought support for foldable phones as well as native support for theming, including the popular dark modes some apps provide. Users now also have access to a native screen recorder and an all new floating settings panel.

Another interesting feature from a user's point of view is the new sharing option: Android 10 provides specific shortcuts to share content directly to contacts.

If your app needs permission to access certain user data, be aware that Google increased security by limiting location access. This adds new permission handling for background interactions and limits the access to device identifiers.

And if you are working on media apps, there are some good news for you: Android 10 supports multiple new multimedia codecs and provides a native MIDI API to interact with music controllers. It is now also possible to capture audio output provided by other apps.

Android 10 allows the developer to provide peer-to-peer network connections e.g. for setups with other devices. To configure things like the network or the volume the mentioned floating settings panel can be triggered by the developers, to grant changing configuration without leaving the app context completely.

An optimized handling for the creation of files on external storage were added, which informs about a pending creation or helps to store specific file types in the appropriate directory.

To provide an enhanced real-time experience while gaming or during other interactions, the Wi-Fi high-performance and low-latency modes can be used. While doing so the new Thermal API helps to keep track of temperature and load of the device.

Hardware Adaptations

Android is not only a smartphone operating system but also a car entertainment system and operating system for wearables and TVs. To adapt to different form factors, controls and usage areas, and to mitigate possible limitations, like e.g. no touch input, multiple variations of the Android system were implemented.

Wear OS

Wear OS[7], previously known as Android Wear, is Google's adoption of Android for smartwatches. It provides a simplified interface, as smartwatches provide much smaller displays than smartphones. Therefore, voice controls are playing an important role in this context. Wear OS is mostly used in combination with a connected smartphone, even if Wear OS devices can technically be equipped with their own SIM card. A key feature is the Google Fit ecosystem of apps that support run and ride tracking, heart activity, step-counting, etc. Users can use their watch to control their phone – music, for example. Notifications via the vibration engine are another key element. Those can be used for notifications from Google Now like flight reminders, traffic warnings, meeting reminders, etc.

---

7   wearos.google.com

Android TV

Android TV[8] provides an adoption of the operating system in regards to big screens and non existing touch interactions. A much bigger interface with simplified controls, usable with remote and voice controls was added. It is also possible to control the TV via the smartphone. One of the most important use cases of Android TVs is the usage of streaming apps like Youtube and Netflix. Some Android TVs also provide direct support for the Chrome Cast Receiver, so users can mirror their screen or stream content directly from their phone or the web to the device.

Android TV apps use the same structure as those for phones and tablets. Developers can thus leverage their existing apps and knowledge to target the TV platform. See *developer.android.com/tv* to learn how.

Android Auto

As smartphones are often used as navigation devices, Android Auto[9] was developed as an adjusted version of Android, which allows to display content on the car's internal screen. Doing this enables a special user interface focused on navigation, communication and entertainment. Interactions are often performed via voice commands to allow the driver to focus on the street.

Software Adoptions

The Android ecosystem allows vendors to extend or change the default behavior and design provided by the system in many regards. Some vendors, like Samsung or Amazon, often provide

---

8   android.com/tv
9   android.com/auto

a completely customized user interface or even different implementations of base functionality on their devices.

This leads to vendor specific ecosystems and different user experiences across the Android ecosystem. This comes with an upside, for instance a very tight integration which provides an amazing experience for users, and also a downside, such as increased fragmentation of the ecosystem.

Google is providing a vendor interface to mitigate inconsistencies, while also providing vendors some flexibility. This should also help to decrease the time-to-update for system updates. The interface is meant to be used by vendors who want to adjust the look and feel of their Android version, but it is not feasible if a total conversion is planned, like Amazon did with Fire OS.

Android One

Android One is Google's attempt to provide a pure Android experience with the latest software updates for two years and security updates for three years. Android One was introduced in 2014 and is meant to be a secure and consistent Android version for everyone. This version can be treated as vanilla Android with some extra optimizations by Google.

Android Go

Android Go is a lightweight version of Android which has been developed to provide a smooth performance also on low-end devices. It is designed to run on smartphones that have 1 GB or less RAM. Even apps on the Play Store will be optimized for these devices mostly targeting emerging markets like India or China. Core features are data saving, less storage requirements and better performance on low-end devices.

Getting Started

The Android operating system is based on free software and uses a Linux kernel as a base. The most important layer for developers is the Java Runtime Environment above the OS layer. The so called Android Runtime (ART) allows the execution of APK files. Those are created by compiling Java or Kotlin code via the OpenJDK for Android. This is the main way to create apps for the Android platform. C or C++ code is also possible via native libraries. Especially resource intense components like multimedia codecs, browser engines, and database encryption layers utilize this approach.

During the Google I/O 2019 conference, Kotlin has been declared the main language for Android development. It should replace Java. However, it is also possible to mix both in one project.

The Kotlin language is developed by JetBtrain, the creators of IntelliJ Idea. It provides a more modern, safe and concise approach to create Android apps. Kotlin code can be compiled to Java byte code and also JavaScript code. Even if Kotlin is progressively replacing Java, Java is still a well-established platform for developing Android apps. But beware, only a subset of the Java libraries and packages are supported and there are many platform specific APIs that will not work with Android. You can find answers to your "What and Why" questions online in Android's Dev Guide[10] and your "How" questions in the reference documentation[11]. Furthermore, Google introduced two sections in their documentation called

---

10   developer.android.com/guide

11   developer.android.com/reference

"Codelabs"[12] and "Courses"[13] that help new developers learning various best practices for Java and Kotlin. This is where you can learn about basics such as navigation and inter-app communication, as well as more advanced features such as intelligent Bitmap downloads and optimizing your app for better battery life. Experienced developers are able to acquire the Associate Android Developer Certification by Google[14].

To get started, you need the Android SDK[15], which is available for Windows, Mac OS X, and Linux. It contains the tools needed to build, test, debug and analyze apps. Development is done within an adapted version of the IntelliJ Idea[16] IDE. This Tool is called Android Studio[17] and allows besides developing, also automatic building, syntax checking and testing.The latest Android Studio versions are bundled with all required tools and components to quick start Android development.

IDE support
Android Studio is the official IDE for Android and comes directly with Gradle Support and many features explicitly tailored to Android development. It is available as pre-packed download including the Android SDK. An extended feature list[18] as well as an end user documentation[19] can be found on the official Android Studio website. Android Studio itself comes with example code and provides code documentation

---

12 codelabs.developers.google.com/?cat=Android

13 developer.android.com/courses

14 developers.google.com/training/certification/

15 developer.android.com/studio#command-tools

16 jetbrains.com/idea

17 developer.android.com/studio

18 developer.android.com/studio/features

19 developer.android.com/studio/intro/index.html

for all system classes and methods available. Android Studio supports also latest Kotlin development features and enables the developer to utilize the modern, safe and concise features of Kotlin.

To measure performance the Android Profiler allows you to keep track of CPU + RAM + GPU usage and the Network Profiler lets you inspect packages send over your wireless connections.

Native development

The Android NDK[20] enables native components to be written for your apps by leveraging both JNI for invocations of native methods and using native subclasses that offer callbacks to its non-native pendants. This is important for game developers and anyone who needs to rely on efficient processing.

---

[20] developer.android.com/tools/sdk/ndk
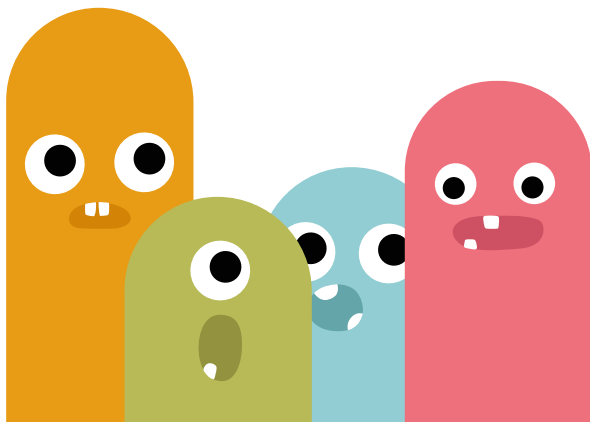
Implementation

App Architecture

Android apps usually include a mix of Activities, Fragments, Jobs, Services, BroadcastReceiver, ContentProvider and more; these all need to be declared in the application's manifest. The manifest also includes the metadata of an application, like the title, version and its required permissions.

An Activity is a piece of functionality with an attached user interface. Fragments can be used to split Activities into smaller and reusable parts. A Service or Job is used for tasks that run in the background and, therefore, are not tied directly to a visual representation. A BroadcastReceiver handles messages, broadcasted by the system, sent by your own or other apps. A ContentProvider is an interface to the content of an application that abstracts from the underlying storage mechanisms, e.g. SQLite.

An application may consist of several of these components, for instance, an Activity for the UI and a Service for long-running tasks. Communication between the components is achieved by Intents or remote procedure calls handled by Android Interface Definition Language (AIDL).

Intents bundle data, such as the user's location or an URL, with an action. These Intents trigger behaviors in the platform and can be used as a messaging system in your app. For instance, the Intent of showing a web page will open the browser. A powerful aspect of this building block philosophy is that any functionality can be replaced by another application, as the Android system always uses the preferred application for a specific Intent. For example, the Intent of sharing a web page triggered by a newsreader app can open an email client or a text messaging app, depending on the apps installed and

the user's preference: Any app that declares the sharing Intent as their interface, may be used.

The user interface of an app is separated from the code in Android-specific XML layout files. Different layouts can be created for different screen sizes, country locales and device features without touching the actual code. To this end, localized strings and images are organized in separate resource folders. Of course, you are also able to define and design layouts in code or make use of both strategies to enable dynamic UI updates.

Android Jetpack

Android Jetpack bundles components for developers to program faster and more robust apps. It contains tools, libraries and multiple other components to simplify development and provide backward compatibility to older Android versions.

Some of the core parts are the foundation components, which provide Kotlin language support and backward compatibility and the Architecture components which help you to structure your app in a proven way. Also important are the Behavior components, those provide easy and stable access to the Android permissions system and the Google Assistant.

The UI components will help you to create more beautiful apps without writing more code.

Every new and existing Android developer should at least have a look if some of the modular designed components, usable via the androidx.* libraries, could be useful for his/her projects.

Material Design

Androids basic design language "Material Design" was introduced in 2014. Designer Matías Duarte explained the basic idea: "Unlike real paper, our digital material can expand and

reform intelligently. The material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch."

Material Design brought a strong, consistent visual identity to the Android ecosystem, parallel but distinct from iOS's flat design and Windows' Metro design. To encourage a solid user experience and consistent appearance of Android apps, Google provides a comprehensive documentation for the design language[21] and a design guide for developers[22]. Going into the importance of color schemes, design patterns, and Material Design, the guide provides a great orientation when building apps for the Android ecosystem.

Material Design also forms the basis for the operating system's UI and the system apps. Additionally, a variety of libraries brought support for Material Design to other different platforms, like the web.

Theming is also supported while still adhering the Material Design guidelines. One example for this is the Basil theme[23].

The SDK and Plug-Ins

To aid development, you have many tools at your disposal in the SDK, the most important ones are:

— **adb:** To query devices (physical and virtual ones) and to connect and interact with them by moving files, installing apps and alike.
— **avdmanager:** To create and manage virtual devices of all types.

---

21 material.io
22 developer.android.com/design
23 material.io/design/material-studies/basil.html

- **emulator:** To emulate the defined features of a virtual device. Due to fast snapshot startups and hardware acceleration the emulator is very helpful for daily development routine.
- **sdkmanager:** To install and update Android SDK components.

These tools along with many others, including tools to analyze method trace logs, inspect layouts and test apps with random events, can be found in the tools directory of the SDK. Most of them are also accessible via the Android Studio IDE user interface.

If you are using features such as ConstraintLayouts[24], be sure to add the corresponding Support Library/AndroidX packages from Google. They are available through the Android Jetpack foundation components. They enable developers to deploy modern features also on older Android versions without compatibility issues. If possible always use those backward compatible components over normal elements to avoid problems on older Android devices. To provide some backward compatibility in general, Google began to use the Google Play Services framework[25] which gets updated via the Play Store and adds some core libraries such as the latest Google Maps version. That is how older devices are enabled to access newer APIs and components of the Google software chain.

Beside backward compatibility it is mostly always recommended to target a recent API version of Android in your build.gradle file, to profit from new system features. Your Android Studio should also be up to date, to be able to use the

---

24 developer.android.com/training/constraint-layout

25 developer.android.com/google/play-services

latest performance improvements and feature updates while developing.

To provide push notifications, analytics, machine learning capabilities and crash reports Google Firebase[26] could be a solution worth investigating. Firebase provides a huge set of functions which are easy to integrate into Android Studio projects.

Testing

The first step in testing an app is to run it on the emulator or a device. You can then debug and analyze it via Android Studio.

All versions of the Android OS are built to run on devices without modification, however, some hardware manufacturers may have changed pieces of the platform. Therefore, testing on a mix of devices is essential. To get an idea of which devices are most popular, refer to AppBrain's list[27].

The Android Testing Support Library provides a collection of modular tools to cover most test scenarios. JUnit4 is the foundation for all unit tests, Espresso[28] provides a very lean API that helps to quickly write procedural tests for your UI and the AndroidJUnitRunner[29] helps you to execute your tests.

Tests can be written using the standard JUnit format, using the Android mock objects that are contained in the SDK. Additional libraries like Mockito can help to increase performance while creating Android unit tests[30].

To automate testing, the Android SDK comes with some

26 firebase.google.com

27 appbrain.com/stats/top-android-phones

28 developer.android.com/training/testing/espresso/

29 developer.android.com/training/testing/junit-runner

30 developer.android.com/training/testing/unit-testing/local-unit-tests

capable and useful testing instrumentation tools. The UI Automator[31] requires Android 4.3 (API level 18) and will present you the captured device interface including some information about the views presented. Executing the tests is relatively easy: After you have written your test, it can be build, executed and delivered to the device or emulator directly from within Android Studio.

MonkeyRunner[32] is a powerful and extensible test automation tool for testing the entire app. These tests can be run on both virtual and physical devices. The tests written in Python can monitor the UI by creating screenshots and sending system events such as key presses. Your tests can then check the status of your app after these events have occurred.

Open source testing frameworks, such as Robotium[33], can complement your other automated tests. Robotium can even be used to test binary APK files if the app's source is not available. Roboelectric[34] is another great tool which runs the tests directly in your IDE in your standard/desktop JVM.

Your automated tests can be run on continuous integration servers such as Jenkins or Hudson. Roboelectric runs in a standard JVM and does not need an Android run-time environment. Most other automated testing frameworks, including Robotium, are based on Android's Instrumentation framework and will need to run in the respective JVM. Plugins such as the Android Emulator Plugin[35] enable these tests to be configured and run in Hudson and Jenkins.

31 developer.android.com/training/testing/ui-automator
32 developer.android.com/studio/test/monkeyrunner
33 github.com/RobotiumTech/robotium
34 robolectric.org/
35 wiki.jenkins.io/display/JENKINS/Android+Emulator+Plugin

Building

Aside from building your app directly in the IDE of your choice, there are also more flexible ways to build Android apps. Gradle[36] is the officially supported build automation tool for Android. There is also a Maven plugin[37] which is well-supported by the community. Both tools can use dependencies from different Maven repositories, for example, the Maven Central Repository[38].

The Gradle build system provides a huge selection of configuration possibilities. It is possible to use different flavors to build e.g. a free and a paid version of the app, while a general differentiation between debug and release builds is also possible. Mostly everything can be configured within your build.gradle file, which enables the developer to also automate the building process.

To shrink the size of apps and to obfuscate the code the new R8 compiler[39] can be used out of the box within your Gradle build pipeline.

To reduce the size of apps even further Google introduced Android App Bundles[40], which contain the compiled code and resources, but generate and sign APKs especially for the requirements of the requesting devices. This happens directly on Google Play if an App Bundle is provided.

36 developer.android.com/studio/build

37 simpligility.github.io/android-maven-plugin/

38 search.maven.org/

39 developer.android.com/studio/build/shrink-code

40 developer.android.com/guide/app-bundle

Signing

Your apps are always signed by the build process, either with a debug or release signature.

The same signature must be used for updates to your app - so make sure to not lose the keystore file or the password. Remember: you can use the same key for all your apps or create a new one for every app.

Google also provides a centralized solution for signing. Google Play App Signing[41] hands over a lot of the work to Google's infrastructure and could reduce problems with managing and securing the needed keys.

Distribution

After you have created the next killer application and tested it, you should upload it to Android's app store called "Play" at *play.google.com/apps/publish*.

You are required to register with the service using your Google Checkout Account and pay a $25 registration fee. Once your registration is approved, you can upload your app, add screenshots and descriptions, then publish it.

Make sure that you have defined a `versionName` and `versionCode` in your `build.gradle` and an icon, a label and the required permissions in your `AndroidManifest.xml`.

The Google Play Store provides, beside production rollouts, also alpha and beta testing plus staged rollouts. This allows you to do some friendly user testing before publishing the app to all users. Furthermore, you can target specific countries and devices by setting the right flags in the Developer Console

---

[41] developer.android.com/studio/publish/app-signing.html#app-signing-google-play

and export detailed statistics that help in understanding your userbase. Using the inbuilt localization service, you can easily add new languages to your app by paying a fee - make sure to check the Localization Checklist[42] for detailed information about the importance of this topic.

As there are lots of competing applications in Android Play, you might want to use alternative application stores. They provide different payment methods and may target specific consumer groups. One of those markets is the Amazon Appstore which comes pre-installed on the Kindle Fire tablet family. But you should keep in mind that alternative play stores force the user to enable unknown sources for app installation, which is always a potential security risk.

And make sure to read the chapter about app store optimization in this guide to pave your road to success.

Monetization

Google Play is the main distribution channel and of course the most popular platform for Android app distribution. Google charges you $25 for the registration and a transaction fee of 30% of your earnings.

But the Play Store is not your only option. For the vendor specific ecosystems, such as Samsung Apps or Amazon's Appstore, you should consider using their SDKs to enjoy the benefits of optimized monetization.

In addition to selling an app in one of the many app stores available, there are several different ways of monetizing an Android app. One suitable way is by using advertising, which may either be click- or view-based and can provide a steady income. Other than that, there are different In-App Billing

---

42 developer.android.com/distribute/googleplay/publish/localizing.html

possibilities such as Google's own service[43] that utilizes the Google Play Store or Square's In-App Payments SDK[44].

Most services differ in transaction-based fees and the possibilities they offer for example subscriptions, parallel payments or pre-approved payments.

Be sure to check that the payment method of your choice, is in harmony with the terms and conditions of the different markets where you want to publish your app to. Those particularly for digital downloads, for which different rules exist, are worth checking out.

Again: There is of course a dedicated chapter about monetization in this guide as well. So go ahead and read further.

43   developer.android.com/google/play/billing/
44   squareup.com/us/en/developers/in-app-payments

Swen Hutop & Dennis Kluge

BY
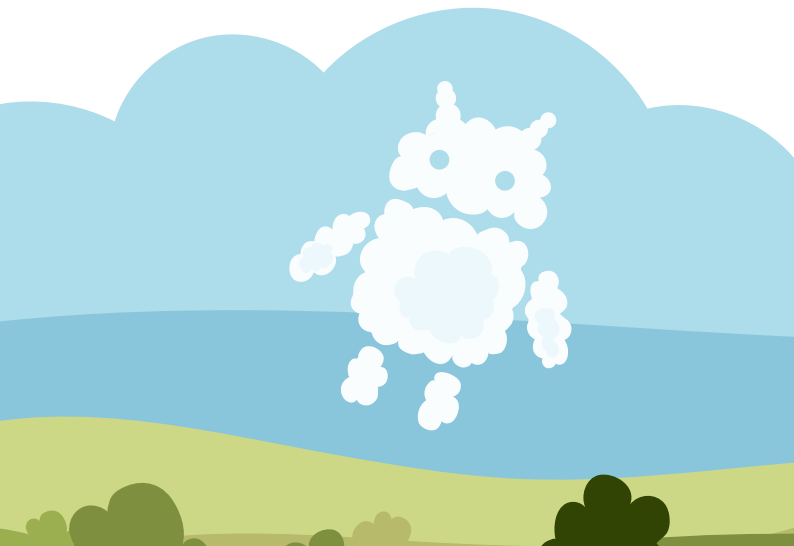
iOS Development

In 2007, Apple introduced the iPhone and significantly changed the way we perceive and interact with mobile phones. Besides the device itself, two other factors made a significant contribution to its success. Firstly, the new iOS operating system, which had been radically designed for gesture-based interaction, and made sure that the hardware and software smoothly merge into one another. Secondly, the App Store has opened up the platform and created the ability to install third-party apps, which led to a thriving market that provides users with a huge choice.

Over time, the original smartphone operating system has undergone an extreme transformation from a once-closed ecosystem. It now includes several forms: iOS for smartphones, iPadOS for tablets, watchOS for smartwatches and tvOS for televisions. All derivatives use the same technical foundation and offer specific elements for creating apps. These applications are developed using Xcode as the development environment.

When comparing iOS with Android, one of its key advantages is the lack of fragmentation compared to Android: Ten months after the release of iOS 12, it has already been installed on more than 80% of devices[1]. In comparison, Android Version 9 only made it to 10% within a whole year after its release[2]. It is not just that iOS versions essentially spread faster, but that the longevity of iPhones is also astonishing. The update period is around four years: Even the iPhone 6s, which was first delivered in 2015 with iOS 9, can

---

[1] macrumors.com/2019/02/25/ios-12-installed-on-83-percent-of-devices
[2] developer.android.com/about/dashboards

still be updated to iOS 13. Another aspect that Apple attaches great importance to, is security and privacy. In iOS 13 for instance, you can use your personal Apple ID, to securely log in to websites, as it is also possible with the Facebook login. The significant difference here is that only randomly generated data is made available to the apps.

Developing iOS apps

First of all, to develop iOS apps, you need a computer running macOS. Apple restricts access to its development tools, since these can only be used on its platform.

Before getting started with coding it is necessary to create an Apple ID and a developer account[3]. If you intend to release the application or deliver it to a beta test group you have to subscribe to the Apple Developer Program. The yearly costs are $99[4].

The architecture of the iOS operating system is built up in different layers[5]. It starts with the Core OS, which handles access to the hardware, sensors, etc. Core Service and Media are intermediate layers which handle e.g. images, video and audio. and lastly Cocoa Touch, which is the primary framework used by developers. It handles the UI creation, Touch Input etc. Each layer adds a certain abstraction, which makes handling the underlying logic much easier.

Choosing the right tool for development is rather easy. With Xcode[6] Apple provides a free Integrated Development Environment which lets you create, test and distribute your app. If

---

3   developer.apple.com
4   developer.apple.com/programs
5   see aruniphoneapplication.blogspot.com/2017/01/ios-architecture.html
6   developer.apple.com/xcode

iOS Development

you are looking for an alternative we recommend to take a look at JetBrain's AppCode[7].

Xcode lets you easily create a new project; multiple templates for iOS, iPadOS and watchOS apps are available. After having chosen your target, you have to provide an App ID. This unique identifier is used to distinguish your app from others. It can be created through the Apple Developer Portal. Furthermore, it is necessary to create keys to sign your application. This process makes sure that the app originates from you, as a verified developer. Nowadays Xcode handles this whole signing procedure automatically[8].

The primary app programming language is Swift[9]. It is a fairly new multi-paradigm language released in 2014 by Apple, which created a lot of traction over the last years. The predecessor was Objective-C[10] an object-oriented interpretation of the C language.

The main pattern used for development is MVC – Model View Controller[11]. However, different approaches like MVVM (Model-View-ViewModel) or even reactive programming are possible. The latter gained more and more traction after Apple's release of SwiftUI and the Combine framework. Both are more explained in a sub-chapter.

A lot of iOS apps are known for their clean and beautiful interfaces. Those can be developed by using Interface Builder[12], a tool provided by Xcode which lets you interactively configure your view. Another way is to declare the user interface within

---

7 jetbrains.com/objc

8 developer.apple.com/de/support/code-signing

9 docs.swift.org/swift-book

10 en.wikipedia.org/wiki/Objective-C

11 en.wikipedia.org/wiki/Model–view–controller

12 developer.apple.com/xcode/interface-builder

your Swift code. Both ways are proven and sufficient. Apple's Human Interface Guidelines[13] provide rules and standards for apps and their interfaces. It is a worthwhile read, especially for beginners.

A huge platform like iOS brings the advantage of a big developer ecosystem. Many libraries are open-sourced and can be integrated into your app. Cocoapods[14] is a tool which lets you integrate third-party code within your iOS project. For example, if you need to add the Instagram API in your new app, there is a pod for it.

Apps can be tested on an iPhone device or through a simulator. Xcode provides a tool which lets you run your iOS code on macOS. You are able to choose between different OS versions and device types. It does not completely replace testing on the device itself, but provides a convenient workflow. It especially reduces waiting time as Xcode has to compile and package the app every single time you want to start a test after a change has been made.

Many aspects need to be covered while developing an app. This incorporates things like unit testing and performance measuring. With XCTest and Profiler, Apple provides tools which already cover these aspects.

We dedicated a whole subchapter about the aspects of distribution and monetization, since many different ways and methods exist. It is important to understand that some restrictions and clear guidelines exist.

Starting iOS development comes with some entry barriers. These include the knowledge of Swift and some general understanding on how UI applications are developed. Fortunately, a huge variety of open resources already exist. Those are covered

---

13 developer.apple.com/design/human-interface-guidelines/ios

14 cocoapods.org

in an extra part. It is worth the effort to become a part of one of the most interesting markets today.

iPadOS

With the first version of iPadOS[15], publicly released in autumn 2019, Apple's tablet got its very own operating system variant - which previously ran the same OS as the iPhone devices. With iPadOS, Apple has succeeded in allowing the iPad to make an enormous leap forward in terms of professional use. First and foremost, multitasking options have been hugely extended. In addition to Picture-in-Picture and Split View, Slide Over functionality also allows switching between apps very quickly. Another long awaited feature is the option to access connected USB hard drives and SD cards directly via the native Files app.

Other important new features of iPad OS 13 include:

Multiple Windows

Multiple window support for iPads[16] introduced the possibility to open several windows of one app simultaneously on an iPad. For example, you can simply move an app icon from the dock to the side of the screen to select existing windows or open a new one.

The system offers two different types of windows: The primary window, which can contain multiple app objects and is used by users over a longer period. And secondly, the auxiliary window. This can only contain a single object of an app. Usually, it only comes in use to perform a single action before

---

15 developer.apple.com/ipad

16 developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/multiple-windows

closing the auxiliary window. For example, in the Mail app, the primary window can be used to display the inbox, whereby the auxiliary window is just used to write a mail.

Drag & Drop

One of the biggest enhancements in iPad OS, besides multitasking, is the Drag & Drop activity[17]. On an iPhone this is only possible within one app. On the iPad, data, text and images can even be moved across apps. You can start as many drag activities as you wish. This greatly enlarges the opportunities available to you for using your iPad productively. The source and destination app for the data to be moved run in parallel and are not paused, so that you can continue to use them. You can also call up the dock and open another app in the split view.

From a developer's point of view, it is possible to develop/ design the app here so that the user sequentially adds Drag Items, i.e. things he wants to move. The target app can then accept and store these items.

Text views and text fields automatically support drag & drop. Collection and table views provide specific methods and properties for doing so. For text views, an API is provided that makes it possible to adjust the drag & drop behavior for the views. Furthermore, each custom UI element can be configured in order to support drag & drop. One of the key points is that developers do not need to worry about the security of moving data between two apps, because iPadOS covers security in this regard.

---

[17] developer.apple.com/documentation/uikit/drag_and_drop

iOS Development

SwiftUI

Apple introduced SwiftUI[18] on their annual WWDC (Worldwide Developers Conference) in 2019. This hails a new approach to developing user interfaces for all Apple platforms. As a consequence, it is now possible to develop SwiftUI user interfaces for watchOS and therefore for the smallest display Apple offers, right up to tvOS for large TV displays.

Therefore, SwiftUI uses a declarative syntax. This means it helps to describe what happens after inputs and UI respond to these actions. This simplifies the code that is written, facilitates understanding and increases readability. Even animations are much easier to integrate in SwiftUI.

Xcode 11 includes new intuitive design tools that allow developers to utilize SwiftUI to develop new user interfaces. Changes made in code can even be directly previewed there. Similarly, changes within the preview directly manipulate the Swift code. This allows programmers to adapt parts of the app dynamically, since they are constantly being compiled and executed. Compared to the current state, this represents an enormous step forward. Previously, by comparison, the app had to be transferred to the iPhone or the simulator after each adjustment to view the changes. SwiftUI and Xcode 11 make it possible for you to open several previews at the same time to test various configurations, such as different text sizes, languages or dark mode.

18   developer.apple.com/xcode/swiftui

Catalyst

The booming mobile device market has led to a decrease in popularity of Apple's macOS desktop operating system for developers. At the same time, the technical foundation for the development of apps on the individual mobile and desktop platforms has become increasingly divergent. The decisive factor for the success of iOS has been UIKit. This technology represents the foundation for the interface experience we have come to expect from Apple. On the other hand, macOS apps are still based on the UIKit predecessor AppKit. This means that iOS apps could only be ported to the desktop environment with a great deal of effort. All of this has led to a decreased popularity of macOS among developers.

During the WWDC in 2019, Apple announced their action plan to address this: Project Catalyst now unites the mobile and the desktop platforms[19]. Catalyst is supposed to make it easy to port iPadOS apps to macOS[20] by adapting the user interface and the interpretation of the inputs. For example, touch gestures are automatically translated into mouse interactions.

As an investment in Apple's mobile platform, this means that new markets and synergies are created with little extra effort. Creating an iPad app, automatically means that a macOS app can also be created. It also implies, that new target groups can be addressed and different monetisation channels can be opened up.

19 macworld.com/article/3402057/swiftui-and-catalyst-apple-executes-its-invisible-transition-strategy

20 developer.apple.com/ipad-apps-for-mac

watchOS

watchOS[21] was presented together with the Apple Watch in April 2015 and is an iOS derivative, especially tailored for this device.

The user interaction patterns and the whole interface have been adapted to the small screen, which does not allow things like multi-finger interaction. This is the reason why Apple has also equipped the watch with a digital crown and a side button. The crown allows users to scroll and zoom within applications. The side button serves as some kind of modified home button known from the iPhone and iPad.

The heart of the interface is formed by the interactive dials. Besides showing the time, they also display information or interaction options desired by the user. These Complications allow access to apps or display data.

Apple offers a variety of different dials. These range from minimalist representations to analog clock interpretations.

The ecosystem allows users to install third-party apps. However, until watchOS 5, those had been bundled together with an iOS app for the iPhone. The reason for this, was that parts of the logic were executed on the smartphone. The iPhone also provided internet connectivity for the first generations of the watch. Since iOS 13, Apple allows stand-alone Apple Watch apps. watchOS 6 introduced a separate App Store for this purpose. This means that the Apple Watch can now be considered a stand-alone, self-sufficient device.

21   developer.apple.com/watchos

Watch apps can be created using the development infrastructure provided by Apple. A special simulator exists in Xcode for this purpose. In addition, WatchKit[22] is provided as a framework.

Amongst other things, it gives access to sensors, sending notifications and many other things. The iOS UI framework UIKit is available on the watch as well.

Apple continues to evolve its smartwatch platform with annual updates. The increasing self-sufficiency of the Apple Watch and its focus on health, have generated a large market. Apple does not announce any sales numbers but it is the most popular watch and Apple even surpasses Rolex in this market segment today[23].

22   developer.apple.com/documentation/watchkit
23   businessinsider.de/apple-q2-2019-earnings-apple-watch-airpods-sales-2019-4
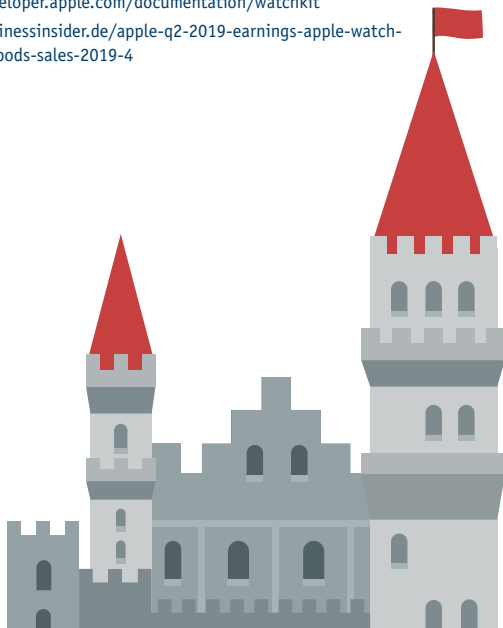
Distribution, App Store & Monetisation

The distribution of iOS apps is done exclusively via Apple's App Store. Users are currently spending approximately $120 billion on iOS Apps per year[24]. However, the gold rush mode of the early days is over, due to a massive competition: There are over 2 million different apps currently available for iOS devices[25].

Uploading apps to the App Store requires a membership in the Apple Developer Program[26]. Two different models exist here. The Apple Developer Program, facilitates the distribution of apps in the iOS App Store to the general audience. With the Enterprise Program, Apple also offers the ability to distribute apps to a dedicated user base only. This is especially important for those companies who want to provide tools for internal usage, only for their employees.

Apple insists on the strict control of the content and functionality of any app submitted to the Store. They are reviewing all app submissions, in order to reduce the risk of providing malware to their users. For you as a developer it means that you should carefully read the App Store Review Guidelines[27] to avoid rejection. And be prepared to wait a little until your app is available to your users, because the review process might take some time. A delay of at least 24 hours needs to be taken in account. Apple provides a separate process for emergencies and serious bugs.

---

24 appannie.com/en/about/press/releases/app-annie-releases-annual-state-of-mobile-2019-report

25 lifewire.com/how-many-apps-in-app-store-2000252

26 developer.apple.com/de/support/compare-memberships

27 developer.apple.com/app-store/review/guidelines

With the help of TestFlight[28], apps and updates can be tested before they are released. Up to 10,000 testers can be invited who can use the application independently from the App Store. However, Apple still reviews these test versions prior to distribution.

The direct monetization of iOS apps is essentially based on three different models: one-time purchase, in-app purchase and subscriptions. To learn more about these options (and other strategies), make sure to read the monetization chapter in this book.

In recent years, Apple has increasingly started to rely on the use of subscriptions. Here, users can subscribe to the app or services within the app.

Apple retains 30% of sales as a fee for all these monetization models. This drops to 15% for subscriptions lasting longer than 12 months.

In September 2019, Apple launched a new, Netflix-like monetization model especially for games. Apple Arcade[29] offers a monthly subscription that gives exclusive access to selected games.

28 developer.apple.com/testflight
29 apple.com/apple-arcade
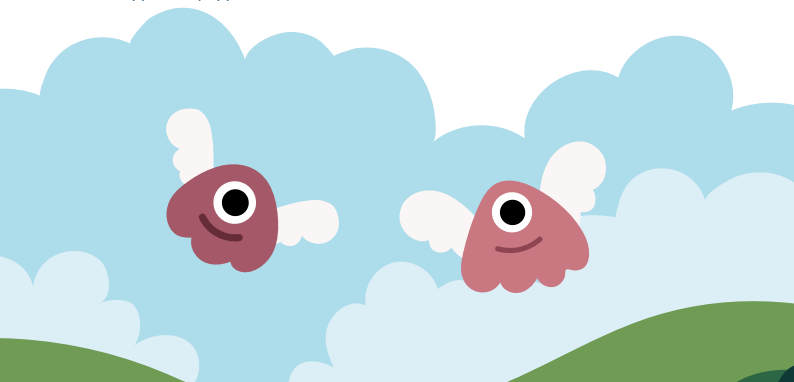
Learn More

The advantage of a mature platform like iOS, is its generous documentation and body of source material. A number of excellent websites exist on the Internet for learning iOS development.

— **Developer.Apple.com** contains complete reference and programming guides for developers to learn how to develop iOS apps and class reference of all classes in their public frameworks. The library website is organized by Resource Types, Topics, and Frameworks plus the ability to search. One important document to read before designing your first app is the iOS Human Interface Guidelines[30]. It contains helpful recommendations on how to design apps to ensure a positive user experience.
— **Swift.org**, the Swift community's official home
— **RayWenderlich.com** has become an essential site for free iOS tutorials written by his community of developers with the goal being "to take the coolest and most challenging topics and make them easy for everyone to learn - so we can all make amazing apps." The site has expanded into offering programming books and video tutorials (with a paid membership). Subscribe to their weekly podcast for the latest news relevant to developers and interviews with leaders in the iOS developer community.
— **iOSDevWeekly.com** is a weekly round up of the best iOS development links. Dave Verwer operates the site, and they offer a weekly email newsletter published every Friday.

---

[30] developer.apple.com/library/ios/documentation/UserExperience/Conceptual/ MobileHIG

- **Merowing.info** is a blog from developer/trainer/speaker Krzysztof Zablocki who offers tutorials and insights into iOS development from his experience as a consultant. He also is active in the Open Source Community, creating tools and libraries for iOS developers.
- **AshFurrow.com** is another popular iOS blogger/developer who proudly states, that the purpose of his blog is "Exploring the Pain Points of iOS." He has authored multiple iOS Development books, is an active speaker and is involved in the Open Source Community.
- **This Week in Swift** is a weekly newsletter involving the most interesting Swift-related news, developments, tutorials and general tidbits related to iOS development.

Summary and Outlook

Since its initial release in 2007, iOS has changed and diversified in many ways. Originally established as a pure smartphone operating system, iOS today runs on tablets, watches and even TVs. Beyond that, the development shows that Apple increasingly wants to establish iOS as a complete alternative to desktop. Precisely, this is being reinforced by the idea of iPadOS.

In addition, the focus on alternative interaction levels is expanding. For example, investments are being made in Siri and the integration of voice UIs in apps. Furthermore, with ARKit, a focus on augmented reality applications can be observed. This creates room for speculation about an iOS derivative for AR glasses, new Apple devices and new markets opening up. Last but not least, it is clear that Apple is increasingly integrating machine learning directly on its devices. CoreML allows developers to easily implement their own AI solutions.

iOS is a vital, and active platform with a mature ecosystem. Join the party.

BY Robert Virkus

Cross-Platform Development

With only Android and iOS as the main players, why should you consider using a cross-platform development framework? In this way even a small team can cater both platforms. And you might even target other form-factors and media easily like PCs, game consoles and websites.

Key Differences Between Platforms

If you want to deliver your app across different platforms you have to overcome some obstacles. Some challenges are easier to overcome than others:

- **Programming Language**: Kotlin & Java for Android, Swift for iOS
- **UI and UX:** Style and interaction patterns differ on each platform.
- **Desktop/Launcher Integration**: On iOS you can only add a badge with a number to your app's icon, while on Android you can add a full-blown desktop widget that may display arbitrary data and use any visuals.
- **Lock Screen Integration**: Again, there are various ways to integrate on the lock screen.
- **Multitasking**: Android and iOS have different options for running background services.
- **Fragmentation**: The Android ecosystem is very fragmented, iOS is a lot more homogenous.
- **Platform Services** such as push, in-app-purchase and in-app-advertisement differ on each platform.

Benefits and Drawbacks of Cross-Platform Frameworks

With the differences above, why should you consider using a cross-platform framework?

— Reusing code reduces complexity and therefore potential bug sources.
— Having mostly one code source also increases time to market and cost-effectiveness of your project.
— You will enjoy feature parity between the platforms.
— Your team needs less platform-expertise, making it easier to assemble.
— Using the same UI components leads to UI consistency that in turn reduces the effort for your support.

However:

— Some features will require native integrations, in that case, you have an increased complexity as you have to deal with your framework, your code and the target OS at the same time.
— Using the same UI may lead to a decreased UX that is not optimal for the target OS.
— Depending on the chosen framework, your code may be more sluggish than native code.
— When there are new OS versions with new features, you may need to wait until your chosen framework has been updated.

Cross-Platform Strategies

This section outlines some of the strategies you can employ to implement your apps on different platforms.

Direct Support

You can support several platforms by having a specialized team for each and every target platform. While this can be resource-intensive, it will most likely give you the best integration and user experience on each system. An easy entry route is to start with one platform and then progress to further platforms once your application proves itself in the real world.

Component libraries can help you to speed up native development, there are many commercial and open source components available for all platforms.

Asset Sharing

When you maintain several teams for different platforms you can still save a lot of effort when you share some application constructs:

— **Concept and assets**: Mostly you will do this automatically: share the ideas and concepts of the application, the UI flow, the input and output and the design and design assets of the app (but be aware of the need to support platform-specific UI constructs).
— **Data structures and algorithms**: Go one step further by sharing data structures and algorithms among platforms.
— **Codesharing of the business model**: Using cross-platform compilers you can also share the business model between the platforms. Alternatively, you can use an interpreter or a virtual machine and one common language across a variety of platforms.

— **Complete abstraction**: Some cross-platform tools enable you to completely abstract the business model, view and control of your application for different platforms.

Player And Virtual Machines

Player concepts typically provide a common set of APIs across different platforms. Famous examples include *Xamarin.com* and *Lua.org*. This approach makes development very easy. You are dependent, however, on the platform provider for new features and the challenge here is when those features are available on one platform only. Sometimes player concepts use a "least common denominator" approach to the offered features, to maintain commonality among implementations for various platforms.

Cross-Language Compilation

Cross-language compilation enables coding in one language that is then transformed into a different, platform-specific language. In terms of performance, this is often the best cross-platform solution, however, there might be performance differences when compared to native apps. This can be the case, for example, when certain programming constructs cannot be translated from the source to the target language optimally.

There are three common approaches to cross-language compilation: direct source to source translation, indirectly by translating the source code into an intermediate abstract language and direct compilation into a platform's binary format. The indirect approach typically produces less readable code. This is a potential issue when you would like to continue the development on the target platform and use the translated source code as a starting point.

(Hybrid) Web Apps

Hybrid web development means to embed a web view within a native app. The standard for hybrid apps is the open source tool Apache Cordova[1] (formerly known as PhoneGap). This approach allows you to access native functionality from within the web parts of your apps and you can also use native code for performance or user experience critical aspects of your app. Hybrid apps allow you to reuse the web development parts across your chosen platforms. Read the web chapter to learn more about mobile web development.

ANSI C

While HTML and web programming starts from a very high abstraction you can choose the opposite route using ANSI C. You can run ANSI C code on all important platforms like Android, iOS and Windows. The main problem with this approach is that you cannot access platform-specific APIs or even UI controls from within ANSI C. Using C is mostly relevant for complex algorithms such as audio encoders. The corresponding libraries can then be used in each app project for a platform.

Popular Frameworks

Here are some popular frameworks that are used to develop cross-platform apps in no particular order.

Xamarin

Xamarin[2] supports both mobile and desktop systems and particularly addresses Windows developer due to its C# and XAML UI support. Traditionally, the UI was created using native

---

1   cordova.apache.org
2   xamarin.com

components, but now generic cross-platform UI components are also supported. Native libraries can be easily integrated.

Cordova

Cordova[3] formerly known as PhoneGap provides an HTML/JS approach that makes it very easy for web developers to create mobile apps. The UI is rendered by the browser elements, so you will have a full web experience. Native code can be integrated using a plugin approach.

Flutter

Flutter[4] is a relatively new framework that renders its own UI completely independent of the OS native UI. Next to mobile platforms, it also supports desktops and the web. Native code can be integrated using a Flutter's channel concept.

React Native

With React Native[5] you code in JS while the UI consists of native elements, ensuring a full native experience. Native code can be integrated using plugins.

Finding the Right Cross-Platform Framework

Here are some questions that you should ask when evaluating cross-platform tools. Not all of them might be relevant to you, so weight the options appropriately. First have a detailed look at your application idea, the content, your target audience and target platforms. You should also take the competition on the

---

3   cordova.apache.org

4   flutter.dev

5   facebook.github.io/react-native

various platforms, your marketing budget and the know-how of your development team into account.

— How does your cross-platform toolchain work? What programming language and what API can I use?
— Can I access platform-specific functionality? If so, how?
— Can I use native UI components? If so, how?
— Can I use a platform-specific build as the basis for my own ongoing development? What does the translated/generated source code look like?
— Is there desktop integration available?
— Can I control multitasking? Are there background services?
— How does the solution work with push services?
— How can I use in-app purchasing and in-app advertisement?
— How does the framework keep up with new OS releases?
— What's the performance of the solution?

Last but not least, for gaming using a cross-platform solution is a no-brainer, as game creation is content-heavy and games do not need to be integrated deeply into each platform.

By Ruadhan O'Donoghue

Mobile Web

While the theme of this book is largely app-oriented, it would not be complete without talking about the mobile web. Indeed the line between apps and web is often blurred in an ecosystem where apps can be built entirely with web technologies, can pull their data and content in via web API requests, or can act as simple app shells for what is essentially a browser (WebView). It can be useful to think of a web-native continuum, with native at one end and web at the other, and various hybrid models in between.

The mobile web and native apps are often pitted against each other as competitors. In many ways they are; often either approach would be suitable to solve a particular problem. It is easy, however, to get lost in the arguments; there are emphatic and obsessive proponents on both sides.

But while apps and web are competing platforms, it is also true that they are complementary platforms, each with its own set of strengths and weaknesses. We will not dwell on the app versus web argument here. Rather we will view them as complementary technologies which often overlap.

That said, with modern web features such as Device APIs, push notifications, installable apps, 60 fps animations, discoverability, the mobile web is a platform both capable and formidable.

Mobile Web Usage

The world has already reached the tipping point where more time is spent on mobile than desktop. And while users spend far more time in apps than on the mobile web, it can be misleading to think that is the whole story. The mobile web has a far larger audience than native apps.

Average Monthly Audience: Top 1000 Mobile Apps vs. Top 1000 Mobile Web Properties
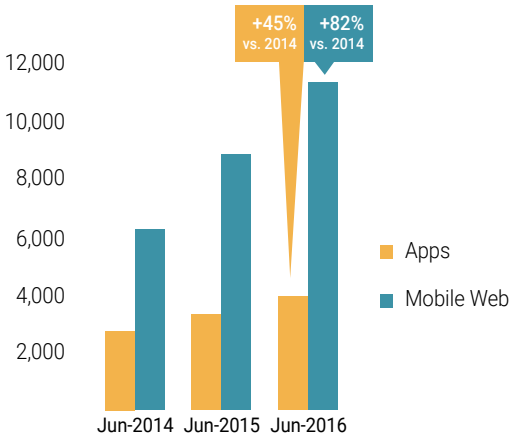
Data from the U.S., users aged 18+, source: *comscore.com/Insights/ Presentations-and-Whitepapers/2016/The-2016-US-Mobile-App-Report*

Most digital strategies will touch the mobile web in at least some way, if not embrace it wholesale.

Devices, Browsers and Fragmentation

The web today is mostly experienced through a browser running on a desktop computer or mobile device. This is where things can get tricky for mobile developers: if you come from a desktop web development background and thought that developing and testing for all the various desktop browsers was hard, then you had better sit down; things are considerably more complex on mobile.

There are at least as many mobile browsers as there are on desktop. But on mobile, in addition to the browser on the device, we also have to consider the type of device and its properties and capabilities. The types of properties and capabilities that can impact web development include

— **Screen size properties** such as physical dimensions, aspect ratio, and pixel density
— **Input types** such as keypads, touchscreens, styli, microphones, and cameras
— **Spatial sensors** such as GPS, accelerometers, compasses, and gyroscopes
— **Network capabilities** such as WiFi, 3G, LTE, and 5G

This results in a staggering number of device-capability and browser combinations that your visitors might be using. If you want to provide a good user experience then ideally you want your web pages to work on all device-browser combinations. This is the scale of fragmentation problems facing web developers today.

What is a Web Browser Anyway?

The web browser is a central part of the web platform. It is a complex piece of software with many roles. It orchestrates the underlying web technologies, combining them into usable web pages. It acts as a window and interface to the web for the user, interpreting the user's actions and inputs, and rendering its response in real-time.

On top of all of this, the major browsers come bundled with a set of complex developer tools, that provide deep insights into the inner workings, structure and performance of the web pages that it renders. There are many developer tool features that help specifically with mobile development. We will see more about developer tools later in the Testing section.

When you build a web page, you are building something to be consumed by browsers, and so you must be aware of their capabilities, idiosyncrasies, and limits, especially on mobile.

Browsers and Rendering Engines

At the heart of every browser is a component that is responsible for laying out and rendering the content of a page. This is known as the rendering engine, or layout engine. Each of today's web browsers is based on one of a small number of rendering engines.

Knowing what rendering engine a browser is based on can help guide development and testing, since any browsers that share the same rendering engine will often behave similarly: they will generally support the same features, and at the same time fall foul of the same bugs.

- **Gecko:** an open source engine used in Mozilla's Firefox browser
- **WebKit:** a widely used engine that powers Safari, as well as all browsers in the iOS App Store, and formerly Google's Chrome. It was built by Apple in 2001 and open-sourced in 2005
- **Blink:** in 2013 Google forked WebKit—which it had been using in Chrome—to create Blink. Blink now powers Chrome, Opera, Microsoft's Edge browser, and all Chromium based browsers
- **Presto:** (retired) formerly used in Opera (now powered by Blink), and is still used in Opera Mini
- **EdgeHTML:** (retired) formerly used in Microsoft's Edge browser (now powered by Blink)
- **Trident:** (retired) formerly used in Microsoft's now retired Internet Explorer browser

What Browsers Should You Develop For?

One constant of the web is the ever-changing browser landscape. Browser popularity will vary from market to market and location to location. You should have an idea of the browser market share in your target market, so that you can prioritize and optimize for these browsers. This said however, you should also try to maximize browser compatibility across the widest range of browsers where possible, since, except for in very limited or constrained circumstances, you will not know in advance what browser or device a user will use.

Worldwide mobile browser market share June 2018-June 2019
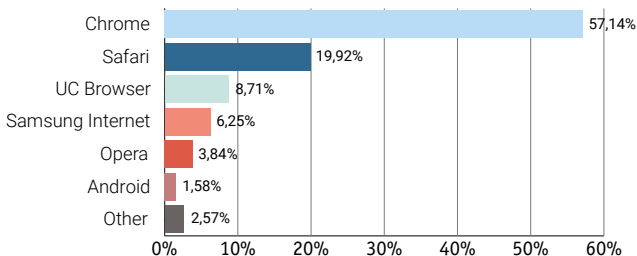
Source: *gs.statcounter.com/browser-market-share/mobile/worldwide/#monthly-201806-201906-bar*

Reports such as DeviceAtlas' Mobile Web Intelligence Report[1] provide periodic snapshots of the device and browser landscape, highlighting interesting market share data such as the most popular OSes, manufacturers, and screen-sizes, and can help guide decisions relating to browsers and browser targeting.

[1] deviceatlas.com/blog/the-most-popular-mobile-browsers

HTML, CSS, and JavaScript: the Building Blocks of the Web

So far we have only covered devices and browsers. Now let us look at the technologies that are used to render web pages for us: HTML, CSS, and JavaScript.

HTML: Structuring Web Content

HTML is the markup language of the web. It is used to structure the content of a web page. Many variations have emerged over the years. Early mobile markup languages include WML, and XHTML Mobile Profile. The most recent iteration, HTML5, has matured sufficiently to encompass semantic markup, custom elements, and device APIs, and is well supported on mobile web browsers.

CSS: Styling Web Content

CSS stands for Cascading Stylesheets and is used to style and lay out web content. CSS rules apply various properties, such as color, to elements by targeting those elements with a selector. Selectors allow you to pinpoint any element or group of elements in a page, so that you have full control: CSS can be used for small styling tasks such as setting the color of text, to large layout tasks that affect the entire page layout.

CSS has gone through multiple specifications, and is split into many modules, each with its own specification, and covering diverse aspects of CSS from selectors to 3D transformations and animations. CSS has become advanced enough that it can often deliver sufficiently interactive experiences without the need for JavaScript. Generally, for performance, you should choose CSS over JavaScript for implementing interactions and animations where possible.

CSS preprocessors such as LESS and SASS are often used to extend CSS with operators and functions and other features that improve the development process and promote code reuse and maintainability.

CSS3 is well supported across mobile browsers.
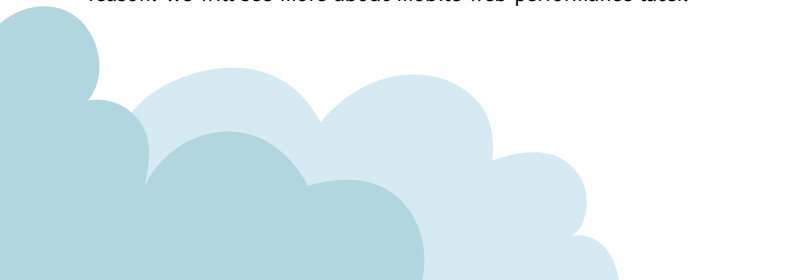
JavaScript: Client-side Scripting

JavaScript adds a programming layer to the web. It is used to bring more complex interactions, animations, and functionality to the web, and can be used to build web applications and games.

Originally only a client-side scripting language, JavaScript has grown and matured over the years as its underlying ECMA-Script specification has evolved. It is now backed by a huge and vibrant community and a wealth of development tools, and has gained respectability as a programming language, and, through Node.js became available as a server-side language too.

Many thousands of JavaScript libraries have been published for developers to use, as well as full front-end frameworks for building entire web apps, such as Angular.js, React.js, and Vue.js.

JavaScript libraries and frameworks can be a major source of page bloat, causing poor performance and bad UX, particularly on mobile. The mobile developer, therefore, should always be wary of adding JavaScript libraries to a project without good reason. We will see more about mobile web performance later.

HTML5

HTML5 is the most recent major version of HTML, and being well supported on mobile, brings many features that enhance the mobile web user experience. But the term HTML5 refers to more than just the HTML markup language; it also refers to related web technology specifications that include CSS3 and many useful JavaScript APIs.

Among the JavaScript APIs included in HTML5 are useful browser features such as the Canvas element, Touch Events, and Web Storage, to name a few. They also include device or hardware APIs that are particularly useful in the context of the mobile web. Device APIs give the browser access and control of specific device hardware features, such as the camera, the accelerometer, and GPS sensor.

HTML5 brings the browser closer to feature parity with native apps—things that were once only possible via native apps are now possible with the web. With location-based APIs web apps can know where the user's device is; through sensor APIs they can access camera images and compass bearings; with storage and caching APIs they can work offline. Websites can even send opt-in push notifications to users, even when the website is not open in the browser. With HTML5, the web experience got a whole lot richer.

Some of the more interesting HTML5 APIs from a mobile perspective are described below.

— **Geolocation API:** The Geolocation API gives access to the geolocation capabilities of a device, which can include precise location data based on the sensors of the device. Location can be based on the available sensors of the device, and includes GPS, A-GPS, Wi-Fi, and cell-based triangulation methods. Latitude, longitude, altitude, heading, and speed data are exposed by the API.

- **Device Orientation API:** Like the Geolocation API this API returns information about a device's physical relationship with the world. While the Geolocation API is concerned with a location in space, the Device Orientation API is concerned with orientation. The information is based on orientation sensors such as compasses, gyroscopes, and accelerometers. The data exposed includes orientation in three axes, acceleration, and rotation rate information.
- **Service Workers:** Service workers allow webpages to run scripts in the background. They can act as a proxy server to web pages, intercepting requests and generating responses, and so they facilitate offline capabilities, background syncing, and push notifications.
- **Push API:** Allows web pages to receive push messages.
- **Notifications API:** Allows a web application to display native-like notifications. Together with the Push API and service workers, web apps can send and receive asynchronous push notifications to a device.
- **Web Payments API:** Keying in credit card payment information has always been a laborious task on the web, but particularly so on mobile where input is more difficult. The Web Payments API aims to solve this problem, while at the same time removing the need to share your payment details with any third-party eCommerce site of unknown trustworthiness.
- **Touch Events API & Pointer Events API:** These are two APIs based around touch screen input, offering information about touches, and swipes etc.
- **WebRTC and media capture and recording APIs:** Allow web pages to interact with a device's media sensors such as microphones and cameras to capture, stream, and record audio and video.

Approaches to Modern Web Development

Modern web development practices must accommodate mobile devices and address the fragmentation issues facing mobile web developers. Several distinct approaches have emerged in recent years.

Responsive Web Design

Responsive web design (RWD) is an approach to web design that delivers resolution-independent web pages—that is, flexible pages that will work well on most screen sizes, from desktop browsers to small screen mobile devices, and everything in between. It is based on three techniques:

1. **A flexible grid:** ensures the page layout scales with screen resolution, rather than using fixed dimensions
2. **Flexible images:** images that will scale within a flexible grid
3. **CSS media queries:** applies CSS rules to distinct ranges of resolutions or device classes, based on breakpoints

RWD became popular because a single HTML page designed this way could be expected to perform reasonably well on a wide range of devices. It is basically a one-size-fits-all solution, which carries advantages and disadvantages:

Advantages

— Resolution independence can mean less time to implement and maintain
— No need to maintain and serve separate versions for different device classes
— Browser features can be detected client-side

Disadvantages

— Only achieves resolution independence, but not a content adaptation, so content is not optimized for all devices
— More bytes than necessary can be sent to a device, which can impact performance
— Can perform poorly, or not at all, on low-end devices since the same content will be delivered to both desktop and mobile device.

Progressive Enhancement

Progressive enhancement is a technique that has been around since 2003[2]. The idea is that you start off with a minimal, base page sent to every device, along with some JavaSript enhancement logic. A low-end device might ignore or fail to execute the enhancements, but will still deliver a functional experience for the user. More capable smartphones, tablets and desktop browsers will execute the JavaScript enhancements progressively until the page is built up to an optimal level for the device.

This approach stands in contrast to the idea of graceful degradation, where rich functionality is built first, and exceptions are added afterwards. This requires additional work to ensure that a page is still functional in the absence of any unsupported features.

In practice, you should consider progressive enhancement as a technique that can be used to smooth over differences in a range of mobile devices, rather than as an overall approach.

---

[2] hesketh.com/publications/inclusive_web_design_for_the_future

Advantages

— In theory, there is no limit to the features that can be
  progressively added
— Can cater for a full range of devices from low-end to
  high-end and desktop

Disadvantages

— Using a single base for all devices can be restrictive
— The actual progressive enhancement JavaScript takes time
  to execute, and can impact performance

Mobile-First Responsive Design

Mobile-first RWD follows the design principles of both RWD and
progressive enhancement techniques. In this approach, the
design begins with a mobile-optimized version and builds up
from there.

Advantages

— Increases reach over pure RWD since it is more likely to
  work on lower-end devices
— Forces designers to focus on content and functionality,
  making it easier to define content hierarchy, with the most
  important content at the top

Disadvantages

— Suffers the same issues as RWD
— May require a complete redesign of existing site, if exist-
  ing site does not follow this approach

Adaptive Web Design (Server-Side Adaptation)

Server-side adaptation[3] uses a device detection solution on the server to map a device's request headers to a database of device capabilities. Once the device's capabilities are known, a page can be built to match the device capabilities resulting in highly optimized pages. Leading solutions include DeviceAtlas[4] and ScientaMobile[5].

Sometimes referred to as "browser-sniffing", the effectiveness of this technique is evident in its adoption by most of the major internet brands that take their web presence seriously, including Google, Amazon, YouTube, Facebook, and Ebay.

Advantages

— Knowing the capabilities of a device means that highly optimized pages can be delivered for multiple device classes
— Extremely reliable and accurate: good solutions report over 99.5% device detection accuracy
— Excellent performance since pages can be fine-tuned for devices

3   There are various definitions of adaptive web design; here we mean that there are some device optimizations taking place on the server.

4   deviceatlas.com

5   scientiamobile.com

Disadvantages

— Involves developing and maintaining different page
 templates for different device classes
— Database of User-Agent capabilities must be updated with
 new devices
— Most solutions are commercial

RESS: A Hybrid Approach

One last approach to consider is RESS (REsponsive design with
Server Side components). RESS combines the adaptive and
responsive approaches to deliver a solution that combines the
best of both. Using server-side adaptation, an initial page
is optimized for a range of devices or device category. Then,
within each category, the content can be adapted further on
the client-side using responsive techniques. This approach
can be pushed further still, so that properties sourced from
the browser can be fed back to the server to further tune the
server adaptation.

Advantages

— Offer the most flexibility and the highest degree of
 optimization of all solutions
— Benefits of high-performance server-side adaptation,
 combined with the ability to tweak with properties
 obtained on the client-side

Disadvantages

— Difficult to implement, requiring device database
— Full round-trip required to get the most benefit

Hybrid Apps

Another class of web application that is worth a mention is hybrid apps. These are often, but not necessarily, built with web technologies, HTML, JavaScript and CSS. Hybrid apps are compiled and packaged as native apps and distributed in native app stores. They are installed like native apps, but are essentially web apps on the inside. Generally, they consist of a full screen webview, or thin native app wrapper and a webview. The webview does the heavy lifting of rendering the web app, while the native library gives access to native APIs and hardware. This is an attractive approach to many since it is possible to leverage web development know-how for native apps without having to learn native platform development. Various hybrid app development frameworks exist to help the developer, including Apache Cordova[6] and PhoneGap[7], and React Native[8]. The most recent framework to enter into this space in 2017 is Google's Flutter[9]. Describing itself as a "UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase", it has become very popular with developers.

It is possible to have an app that is built with web technologies, but that is compiled down to the native code for the platform, for example with React Native. Unless this app is using a webview it is not really a hybrid app, at least in the traditional sense. React Native is essentially pushing the boundaries of what is considered a hybrid app. With React Native apps are compiled into native code for each platform,

---

6   cordova.apache.org

7   phonegap.com

8   reactnative.com

9   flutter.dev

iOS and Android, and widgets are rendered as native platform widgets, while Cordova/PhoneGap apps use use webviews to render web content within a native app shell.

See the chapter about Cross-platform apps for more information around hybrid and cross-platform apps.

Advantages

— Cross-platform: a native app shell uses a webview to render content from a web-based content backend
— Does not require intimate development knowledge of multiple native platforms
— Easier to develop and maintain than distinct native apps for each native platform
— Using a remote backend means the app can be updated without having to resubmit to app stores

Disadvantages

— Performance is not as good as a native app for demanding applications
— UX might not be as polished as a full native app, as compromises are made to satisfy cross-platform constraints

Progressive Web Apps

Progressive Web App (PWA) is a term used to describe web apps that make use of modern browser features to deliver rich app-like experiences. The term was first coined by Alex Russell[10] in 2015 to describe web apps that exhibit the following criteria:

— **Progressive:** they work on all devices, and functionality is enhanced progressively
— **Responsive:** layout is flexible and can adjust to device form factor as appropriate
— **Connectivity-independent:** will function under poor network conditions, and even offline
— **App-like:** feels like an app, with an app-shell, and mostly without full page refreshes
— **Fresh:** pulls in new content whenever possible
— **Secure:** served over HTTPS
— **Discoverable:** identifiable as an app while also being indexable by search engines on the web
— **Re-engageable:** stimulate re-engagement with features like push notifications
— **Installable:** can be added to the home screen of a device

PWAs are possible because the web platform has grown sufficiently mature to deliver such experiences. As mentioned above, several key HTML5 APIs make PWAs possible which support features previously only found in native apps.

---

10 Find Alex' blog at infrequently.org

With the addition of Safari in 2018, PWAs are now supported across all major desktop and mobile operating systems. Windows 10 support adds 700 million active monthly users to the 2.5 billion Android devices, and 1.3 billion iOS devices supporting PWAs.

PWAs can now also be published in each of the main platform app stores. It is more straightforward for the Google Play Store and Microsoft's Windows Store than for iOS. For the iOS App Store, the PWA needs to use a native wrapper such as Cordova before it can be submitted.

Accelerated Mobile Pages (AMP)

Google's AMP project[11] is a publishing format based on an open source web components framework with an emphasis on performance. AMP was originally conceived as a response to Facebook's Instant Articles and Apple's News projects, and so initially the emphasis was on news and blog style content.

However, since its launch in 2015, it has evolved to support a much wider range of content, with interactive features such as carousels, image galleries, interactive menus, and a programming model that supports complex eCommerce features.

Initially, AMP was built as a response to the poor performance of mobile pages and was designed to have <1sec page downloads. Its name reflected this focus on mobile, being and acronym for Accelerated Mobile Pages. As its focus has widened, the AMP team has dropped the acronym, so that the project is simply called AMP, and Accelerated Mobile Pages are no longer used.

11   ampproject.org

What is AMP exactly?

There are three parts that make up AMP:

1. **AMP-HTML:** A flavor of HTML5, which both restricts the tags you can use, as well as adding so new ones
2. **AMP-JS:** A JavaScript library that functions as the AMP runtime, which orchestrates the optimized loading and rendering of AMP pages
3. **AMP-CACHE:** A special cache for AMP pages that enables AMP pages to be rendered instantly in some cases

Getting started with AMP is not too difficult since AMP pages are basically HTML. Every AMP page starts off with some standard boilerplate code that includes the AMP-JS runtime. When the AMP runtime parses the AMP-HTML page and comes across any AMP components, it will inject generated markup into the DOM to replace the AMP-HTML markup; this is what gets rendered in the browser.

Canonical AMP Pages

All valid AMP pages must include a canonical link tag that points to the non-AMP version of the page, if it exists. If there is no non-AMP equivalent, then the canonical link must point to itself. This is a canonical AMP page—a standalone AMP page that serves as both the mobile and the desktop web page. Since AMP supports responsive design and media queries, it can scale responsively to support large or small viewports as needed.

Combining AMP and PWAs

AMP and PWAs have complementary strengths and different weaknesses. It is therefore natural to consider combinations of the two. Several different patterns have been identified that combine the speed of AMP with the richness of PWAs:

AMP as PWA

In this pattern, the AMP page is the PWA. It uses the AMP library, so that a valid AMP page can be served from the AMP Cache, resulting in lightning-fast pages. When links are followed however, the user is brought to the original server, where a service worker can now be used.

AMP bootstraps PWA

In this model, the AMP page uses a special component, `<amp-install-serviceworker>`, to install a service worker in the background on the user's device. The service worker can then bootstrap the PWA by downloading and caching initial parts of the PWA, so that when the user follows a link to the full PWA, it is already downloaded and ready to display.

AMP data embedded in PWA

In this pattern, AMP pages are used as the content backend, within a PWA shell.

The Physical Web

The Physical Web[12], is an open source project that aims to enable quick and seamless interactions with physical objects and locations, via Bluetooth beacons.

Apple has its own, proprietary Bluetooth beacon technology called iBeacon. iBeacon differs from the Physical Web in that iBeacons trigger apps, while Physical Web beacons trigger web URLs. The biggest advantage of the Physical Web over iBeacons is that no app is needed, while an app must be installed prior to interacting with an iBeacon. This significantly lowers the barrier to use of the Physical Web, and increases its reach, without any prior setup, to billions of Bluetooth enabled devices.

Getting started with the Physical Web is remarkably easy. Just get a beacon, set it up to point to a URL, that's it! Users with compatible Bluetooth enabled devices will then receive notifications where they are nearby. This is a simple idea the enables a wide variety of applications, like smart vending machines and targetted marketing in physical spaces.

Google also has a more complex beacon platform[13] with remote cloud management of beacons, which can offer scalability and other benefits for large beacon deployments.

12 google.github.io/physical-web

13 mobiforge.com/design-development/googles-beacon-platform-and-the-physical-web

Web Performance and Why it Matters

Most developers intuitively know that web performance is important; nobody likes to wait for a page to load. But there is plenty of empirical data to prove that performance is crucial, especially on mobile:
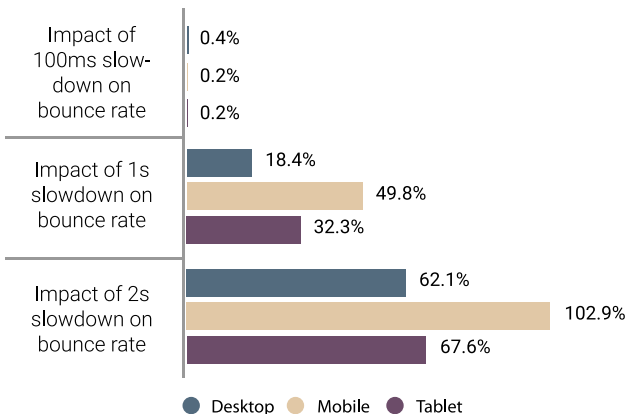
— Walmart experiments found that for each 1 second faster page load time there was a 2% increase in conversions[14]
— Amazon research found that a 100ms latency increase resulted in 1% fewer sales[15]
— A Google study reported that 53% of visitors will leave before a page has loaded if it takes longer than 3 seconds[16]

Depending on which report you read, if your site is slow, you will lose over half your visitors; they just will not wait for it to load, and you will not have a chance to show them what you have to offer. You have a tough audience to please, and the only way to succeed is to deliver a good user experience; and to do this, you will need to deliver a fast site.

To illustrate this, here are some statistics on page slowdowns on bounce rates:

14 webperformancetoday.com/2014/04/09/web-page-speed-affect-conversions-infographic

15 blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales

16 doubleclickbygoogle.com/articles/mobile-speed-matters

Impact of page slowdowns on bounce rates (by device type)

Impact of 100ms slowdown on bounce rate	0.4% 0.2% 0.2%
Impact of 1s slowdown on bounce rate	18.4% 49.8% 32.3%
Impact of 2s slowdown on bounce rate	62.1% 102.9% 67.6%

● Desktop ● Mobile ● Tablet

Source: *blogs.akamai.com/2017/04/new-findings-the-state-of-online-retail-performance-spring-2017.html*

Performance Goals

The idea of a performance budget has been around for a few years; during planning, a "budget" is set on different aspects of how a page should perform, and you try to stick to this budget during development. The specific dimensions of a performance budget might include restrictions on page weight, number of HTTP requests, page load time, time to initial interactivity and so on. If you cannot meet the budget, then you need to consider the assets or features that are blowing the budget. Do you really need that fancy image carousel of JavaScript library for instance?

RAIL

Google defines an approach and set of goals for web performance called RAIL[17]: Response, Animation, Idle, Load. The goals it aims for are:

— **Response:** 100ms—give immediate feedback to user input
— **Animation:** 10ms—when scrolling or animating produce frames in under 10ms to achieve 60fps
— **Idle:** 50ms—non-critical operations should not take longer than 50ms so that application feels fast
— **Load:** 1s—deliver interactive content in under 1000ms to keep users engaged

It is not always possible to achieve all these goals together, especially on low-end devices, so it is sometimes necessary to prioritize these goals, focusing on Load and Response first for example.

Analytics

Analytics is vital to understanding your visitors and traffic. It can be particularly useful on mobile to help you understand what devices your users are using. For many, however, analytics starts and stops with installing a Google Analytics script. But Google Analytics is not the only show in town.

Analytics tools can collect their data on the client or on the server. It is worth noting that relying solely on JavaScript based analytics can be problematic, especially on mobile. If a device fails to run the analytics script—for example, if it is an older device—then you will have no visibility of this device at all, and it can lead you to focus on the wrong devices. Additionally, many ad blockers also block client-side analytics such as Google Analytics. If you are more serious about your analytics, a more accurate picture of your data can be gained by employing a combination of both client and server-side analytics.

Popular tools include Google Analytics[18], KISSMetrics[19], and Matomo[20]. Some tools, such as wao.io[21] provide both server and client side analytics.

18 analytics.google.com
19 kissmetrics.com
20 matomo.org
21 wao.io

A/B Testing

A/B Testing is a very useful technique that can be used in web development to evaluate the performance of alternative interface design, messaging and layout variants so that you can make informed decisions about the UX you deliver. Most of the main analytics tools support A/B testing, including Google Optimize, Google Analytics (Experiments), and Matomo.

Real User Monitoring (RUM)

RUM involves continuous monitoring of user interaction with a web application in real-time. This allows the website owner to quickly pinpoint, prioritize and remedy issues with availability, functionality, responsiveness, and so on. This, in turn, gives valuable insights into user behavior and satisfaction and enables you to fine-tune and improve the overall user experience. RUM can take place client-side or server-side and plug-in tools such as Sevenval's wao.io can correlate front-end and back-end metrics without having to make any code changes to the application.

Monetization

Ads

Ads have traditionally been one of the most common ways to monetize a website, and no less so on the mobile web. There are plenty of ad networks to choose from.

Ad blockers

If you choose an ad-based monetization model, note that there has been a growing backlash against ads—particularly on mobile—since Apple added support for ad-blockers into Mobile Safari in 2015. Ads have a bad reputation for adding unnecessary page bloat and degrading web performance. Mobile and desktop browsers are increasingly shipping with built-in ad blockers and enhanced privacy protection. Even Google, a company whose main revenue stream is based on ads, has included an ad-blocker in its Chrome browser. Therefore, if you are relying on ad revenue, be sure to know the risks and potential downsides.

When choosing an ad network, pick a reliable one that will not harm the performance of your site. A relative newcomer to the field is AMP for Ads (A4A)[22]. These are ads that are based on AMP technology, but will work fine for desktop too. A4A has strict rules about what is permitted and performance-degrading ads will be removed from a page, so you can be confident that ads do not cause UX or performance issues.

---

[22] See ampproject.org/learn/who-uses-amp/amp-ads and github.com/ampproject/amphtml/blob/master/extensions/amp-a4a/amp-a4a-format.md

Ecommerce

There are many ways to build a web eCommerce solution, ranging from off-the-shelf solutions to custom developments. Many new and traditional payment service providers, such as PayPal and Stripe, have good mobile support and are relatively simple to implement. Additionally, a new breed of NFC-enabled mobile wallets, such as Android Pay, Apple Pay, and Samsung Pay, can increasingly be used on the web.

Payment Request API

One final technology to watch is the Payment Request API. This is a recent HTML5 specification that aims to provide frictionless payments on the web, in two ways

1. Removing the need for cumbersome input of payment details
2. Removing the requirement to share your credit card details with third-party sites of unknown trustworthiness

Several browsers already support the Payment Request API, including Chrome for Android, Edge, and Samsung Internet, and it is available in developer (nightly) builds of Firefox.

Please see the Monetization chapter for more detail on the types of monetization models that can be used.

General UX and Performance Guidelines

We do not have the space here to go into much practical detail here[23], but a general set of guidelines for mobile web development includes the following.

UX

— Optimize for mobile
— Do not require pinch-to-zoom
— Make product images expandable
— Ensure tap targets and links large enough for "fat" fingers
— Keep calls to action front and center
— Keep menus short
— Intuitive navigation with a prominent link to the homepage
— Avoid interrupting pop-ups and large interstitials
— Include a site search, with filters to narrow results where possible
— Include click-to-call where possible
— Only ask for special browser permission when needed, not up-front. E.g. Ask for permission to send push notifications when a user has already indicated a desire to subscribe, or say for updates on an order, and geolocation only when needed for mapping or address.
— Let users browse as guests
— Let users purchase as guests
— Autofill form inputs where possible

---

23 Check this site for some graphical examples: developers.google.com/web/fundamentals/getting-started/principles

Performance

— Keep page weight small, avoid unnecessary media such as images and videos
— Compress all images
— Avoid redirects
— Keep the number of external resources low, to reduce HTTP requests
— Implement caching
— Lazy load images and content as needed
— Minify text resources
— Avoid JavaScript and CSS frameworks unless necessary
— Avoid unnecessary embeds and includes
— Use ads responsibly, and only use lightweight ads
— Define a performance budget, and try to stick to it

Testing for the Mobile Web

Testing on the web is crucial. And on the mobile web, while simulators, and even desktop browsers are useful, the most reliable testing is that performed on real devices. Earlier we talked about device fragmentation. This, combined with the variety of browsers available on each platform makes for a headache when it comes to testing your web pages. It is next to impossible to exhaustively test on all browsers and devices. None but the biggest and best funded projects will be able to even come close. So, when it comes to testing the mobile web you need to prioritize.

If your budget can afford it, it is recommended to acquire several devices on each of the main mobile platforms, Android, iOS, and Windows. Devices should be a mix of high and low-

end, and with the ability to switch between mobile networks if possible.

Broadly speaking, testing effort is divided between functional and UI testing on the one hand, and performance testing on the other. Functional and UI testing is concerned with testing business logic, user interface, UI components and usability. Sometimes issues will only show up on some devices on some platforms, and this makes testing on mobile difficult.

Performance testing is more concerned with how well the site works: is it fast, does it feel fast, does it work under poor network conditions?

Manual testing will likely be the first step in any web testing. While there are great efficiency gains to be achieved through automated testing, automated testing is not always practical, perhaps due to the small size of a project, or time or budget limitations.

Thankfully, there are plenty of tools to help with all aspects of mobile web testing.

Browser Developer Tools

Even for mobile web first pass testing is often carried out on a desktop browser with its built-in developer tools. Just some of the features of a browser's developer tools include DOM inspection, network inspection, performance profiling

— **DOM inspection:** allows examination of the HTML elements that make up a page
— **Network throttling:** for simulating slow and poor network conditions, such as Edge, 2G, 3G and so on
— **CPU throttling:** to simulate high-end or low-end smartphones
— **Waterfall chart and timelines:** provide visualizations of how a page loads and renders over time

— **JavaScript debugging:** allows the developer to examine, add breakpoints, view events, and step through JavaScript code
— **CPU and Memory profiling:** records how the CPU performance and RAM footprint over time

Other features particularly useful for mobile web development and testing are

— **Responsive design mode:** simulates viewports at a variety of configurable sizes, so that you can see how the UI behaves under differently sized viewports
— **Screen mirroring:** a connected device's browser is mirrored to the developer tools and can be interacted with via the desktop browser

Remote Debugging

All of the major mobile platforms support remote debugging of mobile devices. Remote debugging allows you to attach a mobile device to a desktop machine and apply the developer tools of that machine's browser to test and profile the web pages on the mobile device.

Remote debugging is an extremely useful tool, since it allows you to test on real devices, and on real networks. Of course, you still need devices to test with, and that can get expensive. At the very least, you should be looking at having a low-end and high-end device on each of the main mobile OSes: Android, iOS, and Windows. Even then, there will be major gaps in your testing coverage; this is where device labs can help.

Performance and UX Testing Tools

Selenium WebDriver

Selenium WebDriver[24] is the leader in automated web testing. Automated testing is very useful for quickly finding issues with user interfaces, and can be used for regression testing to quickly find breaking interface changes.

Selenium additionally supports mobile testing[25] on Android and iOS, and both simulator and real device testing are supported.

Webpage Test

WebPagetest[26] is an open source and free-to-use performance testing tool which offers remote testing on real desktop and mobile browsers at different locations around the world. It provides waterfall performance charts, as well as measurement of key performance metrics such as time to first byte, speed index, and a number of DOM elements.

mobiReady

MobiReady[27] is a free tool for developers, designers, and marketers that tests web pages and sites for mobile-readiness based on mobile web best practices and standards. It returns a detailed analysis for a page, and offers recommendations on how to address any detected issues. It also includes:

24 seleniumhq.org/projects/webdriver
25 github.com/SeleniumHQ/selenium/wiki/WebDriver-For-Mobile-Browsers
26 webpagetest.org
27 mobiready.com

- device visualizations on low, mid, and high-end devices, showing how the page will look on a variety of screen sizes
- breakdown of page weight for each device class
- a benchmark report of how your page scores against the Alexa top 1000 sites

MobiReady also exposes an API that can be used for automated testing of entire sites.

Lighthouse
Lighthouse[28] audits a web app for PWA features, including:

- can it load offline or under poor network conditions?
- is it fast?
- is it served from a secure URL?
- does it implement accessibility best practices?

Lighthouse is available as an online service, a Chrome extension, a command line tool, and is integrated with Chrome developer tools. The command line version is useful for automated testing.

PageSpeed Insights
PageSpeed Insights[29] is a tool from Google that measures page performance for mobile and desktop visitors. It checks for common performance best practices, and ranks pages out of 100. When issues are detected, it offers advice on how to fix them.

---

28 developers.google.com/web/tools/lighthouse/
29 developers.google.com/speed/pagespeed/insights

Device Labs

The idea of a Device Lab has been around for many years. A device lab is simply a collection of devices that can be used for development and testing. Device labs fall into two categories: physical and remote.

Remote labs offer the most convenience: you install a client on your computer and it allows you to access the real device remotely over the web. On the other hand, with a physical lab, you can hook a device directly up to your laptop and use the browser's remote debugging tools. This can be helpful for solving issues that affect specific devices.

AWS Device Farm

Amazon's AWS Device Farm[30] is an advanced device testing lab that features automated testing against a large collection of real devices in the AWS cloud, as well as direct remote access that allows you to interact with swipes and gestures in real-time from your web browser. AWS Device Farm contains a wide variety of new and old iOS and Android devices.

AWS Device Farm supports pay-as-you-go and flat-rate charging. On the PAYG model, the first 1000 minutes are free, so it can be a good way to try the service before committing to it.

BrowserStack

BrowserStack[31] offers remote testing on a variety of desktop and mobile browsers, and different operating systems. The mobile devices have been chosen for "maximum market cover-

---

[30] aws.amazon.com/device-farm

[31] browserstack.com

age" and includes a vast array of real iOS and Android devices. Supports automated testing via Selenium cloud testing. Browserstack is free for open source projects, and offers a free trial for commercial projects.

Samsung's Remote Test Lab

Samsung's Remote Test lab[32] offers free remote testing on Samsung devices. You reserve and connect to real devices through your web browser. The device list includes old and new phones, tablets, and watches—not surprisingly all Samsung devices—and spread across its Galaxy, Z (Tizen), and Gear brands.

SIGOS App Experience

SIGOS App Experience[33], formerly known as Keynote Mobile Testing, formerly known as DeviceAnywhere (it is hard to keep up!) was one of the first Virtual Device Labs. Despite its latest name, SIGOS App Experience can be used to test on the web as well, on its more than 2000 devices. SIGOS supports both manual and automated testing. The service is accessed via a desktop browser. Offers a 7 day free trial.

Perfecto Mobile

Perfecto Mobile[34] offers paid-for remote testing on real devices. Supports manual and automated testing on multiple devices. A free trial is also available.

---

32 developer.samsung.com/rtlLanding.do

33 appexperience.sigos.com

34 perfectomobile.com

Open Device Lab

The Open Device Lab[35] is a community approach to device labs. Participating organizations offer a physical space where developers can go and use for free. There are currently 154 Open Device Labs across 35 countries offering free access to 4255 devices.

Resources

— Resource site for mobile web design and development: *mobiforge.com*
— Browser feature support and compatibility: *caniuse.com*
— Responsive website design with RESS: *smashingmagazine. com/2013/10/responsive-website-design-with-ress*
— Your first Progressive Web App: *developers.google.com/ web/fundamentals/getting-started/codelabs/your-first-pwapp*
— Offline Web Applications: *udacity.com/course/offline-web-applications--ud899*
— The AMP project: *ampproject.orgs*
— Progressive Web AMPs: *smashingmagazine.com/2016/12/ progressive-web-amps*
— The Physical Web project: *google.github.io/physical-web*
— A good perspective on software quality and testing using web browsers: *mobiletestingblog.com/2017/05/01/recent-web-browser-quality-related-innovations*
— High Performance Mobile Web - Best Practices for Optimizing Mobile Web Apps, by Max Firtman (O'Reilly Media, 2016)[36]

---

35 opendevicelab.com
36 available via shop.oreilly.com/product/0636920035060.do

Mobile Gaming

The Mobile Gaming Economy

Mobile remains one of the most exciting platform for game developers today, at least when it comes to revenues: Since 2014 it is the largest sector of the video games market estimated to represent 45% of total global revenues in 2019[1].

At the same time, games are the dominating category in the app stores when it comes to revenue: Games accounted for 75% of mobile app revenue in Q2 2019[2].

In terms of content and brands, the rise of games like Fortnight and PUBG as well as classic PC games like Old School Runescape and Roblox have started to challenge games like Clash of Clans and Pokemon Go! At the same time titles like Homescapes and Matchington Manor have started to challenge King's dominance of the casual market. There has also been a rise in the number of second (and third) generation teams coming into the market with great success such as Futureplay and Next Games. Despite this, the challenges of becoming successful in mobile games continue to increase as user acquisition becomes more costly.

Making games work on multiple platforms has become easier to do. Looking at the top 1000 free mobile games in 2017 we found 41% were made by natively or using in-house tools; 34% used Unity; 18% used Cocos2D; 2.6% used Corona, 2.2% used Unreal; 1.2% used Marmalade and 0.7% use other tools such as GameMaker, V-Play, etc. Each engine offers different

---

1   newzoo.com/key-numbers newzoo.com/key-numbers

2   appannie.com/en/insights/market-data/q2-2019-mobile-games-market-index

advantages and perspectives to enable developers of different skill types to quickly realize their ideas and prepare them for release.

The time when a random indie-developer can strike it rich with crazy numbers of users is essentially over. Instead, the mobile games market has become a sophisticated space with many different facets and challenges and most games will still fail. Before you start creating your game you need to pay attention to understanding the nature of the market and audience. An essential part of this is that it has become an enormously competitive market, with vast numbers of small teams producing huge volumes of content and spending millions on development and advertising to retain their positions.

Making The Right Game

Creating delightful experiences for your target audience requires just as much creativity as ever, perhaps more. Different developers have different approaches. For some it starts with an emotion (how do you want the player to feel); for others it is about taking a game they love and realizing it in their own unique style; there are also those who start with a mechanic or an art style and the gameplay evolves from there. Making games based on existing titles sounds easy, but in practice rarely succeeds unless you bring something special that has not been done before. Search for sideways scrolling platform games using pixel art and you will find out just how saturated that market is. Doing something brand new comes with its own risk too as players need something of the familiar to help them understand and relate to your innovations. Scott Rogers in his book "Level Up" described this as the "Triangle of Weirdness"[3].

---

[3] mrbossdesign.blogspot.co.uk/2008/09/triangle-of-weirdness.html

He claimed that games consist of a world, activities, and characters. We can change any of these for new ideas, but we cannot change all three without risking losing the audience.

For me, the kind of fun we are looking for in games is something which happens when the player is able to suspend their disbelief and engage in an experience which is free of real-world consequence. We become totally absorbed in the mechanics and narrative of the experience. Curiously, challenge and frustration are as much the motivations to play as they are potential causes to leave the experience. If we achieve a balance between these states of mind we attain a joyful state that all game designers know about, 'Csikszentmihalyi's Flow'[4].

We have to pay attention to what makes a game fun specifically on a mobile device, is different from what we do on other platforms. Generally, mobile games tend to be simple and accessible with enough depth and a sense of purpose and progression to retain player attention. But it is not that easy. The phone is our most personal device and that affects the way we play. There is also already so much substitute; games have to stand out and emotionally connect in order for players and app stores to select them. Part of the game design process is to create enough anticipation to encourage the player to download (even if free) and communicate the aspiration that spending money will make the game even better.

The type of game you make matters and for those unfamiliar with the principles, some are developed from emergent mechanics, with building blocks which combine to create surprising or strategic outcomes, like Chess or Clash of Clans. Then there are those games built using a series of progressive decision points each resolved with their own steps chained

4   scienceandvalues.wordpress.com/2010/02/26/csikszentmihalyis-flow-pleasure-and-creativity

together to make a story such as Episodes - Choose Your Story or the classic Monkey Island. We can even build games which incorporate player creativity such as Roblox or Minecraft, or simple abstract puzzles like Merge Dragons or Origame. Whichever path we take, balance is at the heart of our thinking as a developer. We have to decide how much the game will be affected by skill and how much by luck, the extent to which the game follows a fixed narrative or is player led and of course the complexity of the internal systems, whether that is about character development or a resource economy. With Free-To-Play we also have to consider the impact of money spent on the game experience.

What ever the kind of game is vital that the developer focuses on what matters to the player. The most important questions we should ask is why the player should care. We need them to not just install, but to make our game their distraction of choice. Mobile play may at times start out as a distraction, but in the end, we often spend more time playing on our phones than our consoles. With some much competition, we have to ask why would they play your game? We have to be able to answer that question honestly.

Engaging the Mobile Player

When we develop mobile games we are creating an experience to entertain players on a specific kind of device. Tablets and phones fulfill different needs and require an attention to detail on the specific and different mode of use they fulfill. The phone is generally about "the next minute", it is something we get out when we expect something to happen or need something to occupy ourselves - and increasingly played in Portrait as players are super reluctant to turn their device on its side to play in landscape. Those players that still use their tablet devices will tend to play for a longer period of rest or

relaxation. What does that mean for the game we want to make?

Part of the appeal of any game is tied into its distinctive vision, visual style, compelling gameplay narrative and how the experience is designed to affect the emotions of the players. This has to be appropriate to the nature of how the game is consumed and in mobile, we have to understand the restrictions inherent to these devices. The limited screen size, touch screen controls, accelerometers, battery life, the ability to be interrupted, the ease players have to get out and put away the device, the limited speaker quality, the high-quality headphone output, etc all affect the way players interact with the device. Mobile phones are (mostly) connected to the internet and are the most pervasive of devices as we always carry them.

To show you what I mean, think about the way we implement controls. Touch screens allow a huge range of movement on a 2D plane, but after a short while our skin will get hot and lose capacitance which means the controls will become less reliable. If we simply try to duplicate a twin stick control system (like too many games have) we will soon find problems with the playing experience. That being said with the rise of first person shooters now on mobile, it seems that many players are finding their own way through the limitations. However, there are still plenty of opportunities to design game mechanics with controls specifically to make the touch process feel good or which recognize the limits of the methods available to us and make that part of the experience.

Simplicity remains an important factor for any design and is supremely difficult to achieve because even here we have to balance accessibility with the ability to keep people playing over time. This thinking has spawned its own sub-genre over the last couple of years as HyperCasual games have thrived reducing gameplay to a single continuously repayable mechanic.

It is useful to think about the game mechanic as separate from the context - the part of the game which gives the player a reason to repeat the mechanic. Mechanics are the bare bones of any game. The core loops of actions with starting conditions, challenge, resolutions, and reward. These are the actual things players do in the game. The context is the reason to do that again. This can be a narrative, level structure or even simply a way of gating access to content by player performance. But it has to deliver a sense of purpose and progression. More than that the context needs to call for the player's attention after their session has ended, sitting in the back of our mind as 'unfinished business' enticing us to return later for another session.

Games like CSR and Candy Crush introduced this method of game design to the mobile market. Finding ways to chain together a series of grinding mechanics to sustain playability over thousands of sessions whilst always giving a sense that the goals are achievable is magical. It builds long term engagement and keeps players involved with your game ever longer, provided the experience is sufficiently meaningful. Keeping players playing longer has a direct impact on their willingness to spend money in the game. In a 2014 survey by Unity[5], the reported spend by paying players who spend less than an hour in game averages at $0.66, but for those who spend over 10 hours, this rises to $15.15.

5   gamesindustry.biz/articles/2014-10-14-mobile-spending-driven-by-35-44-
     year-olds

Hypercasual game introduced a more 'disposable' form of gameplay offering captivating repeatable experiences within throwaway games, allows players to restart quickly, watch lots of ads, and move on the the next title. However, sustainable revenues for a single title still seems to rely on managing the player life-cycle.

Designing the Player's Journey

The realization that long term engagement matters has a profound impact on the way we look at game design. The idea of a game as a mechanic or story is transformed to the realization that it is not only our hero character who is going on a journey but our player as well. This journey consists of several stages:

1. Discovery

Players have a particular set of needs and aspirations when they first encounter your game and there is really very little beyond the icon and the first sentence of your app store listing to motivate players to download and play the game the first time. Despite that setting the right expectations is essential. If the game charges upfront, let the player know why they should still buy it, and what they will miss out on if they do not. If the game is free we still have to create expectations, but we also have to show the player why the advertising is worth the hassle or if there are in-app purchases, why those players will feel good about buying them. This is a delicate art.

2. Learning

Once they have taken the choice to install your game we have to make it as easy as possible to engage. Make the icon and name of the game instantly recognizable and ideally a tease, a reason to kick off the app. At this stage we do not want them

to make choices about what characters to play or what levels they want – they do not know yet. Do not make them sign up to Facebook or set-up an account before playing – show them what the game is all about. Then knock their socks off! I often talk about "The Bond Opening", comparing this first ever play of the game to the opening 5 minutes of every Bond movie. It blows us away and at the same time set up everything we need to know about the story, super-secret agents and the world the story takes place in. But it does more than that, it ensures we never want to leave the seat. It is this dual role of showing not telling that sets up the expectations for the rest of the movie which applies to games so well. As this is a game, we do not want to show or tell, we need to "Do!". Players need to learn about games by doing and feeling good about their achievements quickly.

3. Engaging

If we succeed and set up the right expectations to keep them playing we really start the process of building long term engagement. At this point the player understands the challenge and progression and is already returning to continue to play for subsequent sessions. At this stage they should also understand the value of investing further time or money into the game. It is much easier to sell IAP and leverage the use of Opt-In Ads to players who already get the benefits of the game. Sustaining this over the longer term however is challenging. We need longer term 'achievable' goals as well as events and social engagement if we are to keep players, and we have to do all this without over-complicating the game.

3.1 Potential To SuperEngage

Players are not all created equally when it comes to their desire to spend money in your game. Some players become

'Super-fans' who actively desire more and more items to enhance their playing experience. The game becomes their 'hobby' and something they may be willing to invest beyond the price of a cup of coffee. This generally happens only after they fully engaged and where the developer is able to focus on delivering value for that player. It is important not to confuse these players with people who are addicted. Addiction is where individuals have a compulsion which overwhelms their otherwise rational behavior. In practice the majority of true fans are rational people who have made your game their principle hobby. Addictive behavior is always detrimental to the individual and we should do everything we can to help anyone with these kinds of issues.

3.2. Re-engagement

Players who have become engaged will still have a lifecycle; but it may become possible to re-engage them when you add a new update, new content or even events inside the game.

4. Churning

The final lifestage we have to acknowledge is "Churning". It is inevitable that in the end players will stop playing our game. We want to delay that as long as possible, but to fail to plan for that is going to cause us more problems.

Analytics and Game Flow

Making a game is a little like designing an experiment; especially in this data rich, connected era of lean development and minimum viable products. We make a hypothesis and test it as simply as possible. We want to know as fast as possible if we are on to something or not, ideally before spending a lot of money on unnecessary development. That means we need analytics to help us understand what is going on at every life stage.

First, we do not have to capture everything. There are some kinds of data which are static, reference information. For example, the specific position in a specific map. As long as the version of the map used at that time is known then X,Y,Z coordinates alone can be used to create a heat map later. We can also infer a lot of data from other events as long as there is some connecting information. For example, we do not need to capture the level that the player is using for that game in every event or even a list of all the players in that session. We can capture that information with a specific 'Start Session' events and use the associated session ID to allow us to identify everything that happened in that specific game session. Most commercial analytic platforms will automatically capture common data sets like date/time, X,Y,Z etc into their event collection process.

We also have to understand that the data we collect will always be incomplete. For example if the battery dies or the player switches to a phone call – we will probably not get the last upload.

We have a duty to treat player data very carefully, we need to make sure that players remain anonymous. We do not want or need to spy on our players but we do need to understand how the game plays across all players without falling foul of

data protection (especially related to children). In case of any doubt make sure you get qualified legal advice.

What Events Should We Capture?

When considering which events to track, think of them in terms of the timeline in which the player might encounter them. There will of course be a first-time player experience, but it can also be useful to map out more commonly experience 'Engaged player' session flow. We are not necessarily mapping every button press directly. Instead you should look for moments where there are meaningful choices. There is an approach used by the food industry called HACCP[6].

Some typical events worth tracking might include the following:

— **GameMenuLaunch**
AnonPlayerID; TimeIconLaunched

— **SessionLaunch**
TimeSessionLaunched; AnonPlayerID(s); SessionID; LevelIDSelected; OptionSelected

— **SessionStart**
TimeSessionStarted; AnonPlayerID; SessionID;

— **ObjectiveSet**
TimeObjectiveSet; AnonPlayerID; SessionID; ObjectiveID;

---

[6] develop-online.net/opinions/navigating-the-hazards-of-game-data/0187815: In essence the point is that we are looking for the 'Hazards' in the flow of the player experience such as whether they churn (i.e. leave the game) but also trigger points for more positive action such as paying for an IAP or watching a video ad.

— ObjectiveMet
TimeObjectiveMet; AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation

— TargetHit
TimeTargetHit; AttackerID(AnonPlayerID?); SessionID; TargetID(AnonPlayerID?); Damage, XYZLocation

— PlayerDeath
TimePlayerDeath; AnonPlayerID; SessionID; XYZLocation

— LevelComplete
AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation

From creating events in this way we can infer a huge amount of information. For example, if we want to know the percentage of players who complete a level we can count the number of 'GameMenuLaunch' events with the number of 'LevelComplete'. But we can also get smarter with our analysis. We can look at how many people completed a specific ObjectiveID in a specific 'LevelIDSelected' and compare that to the number of 'LevelComplete' in the subsequent level to find out if skipping objectives in earlier levels have a particular impact on performance later.

Once we have the data we need to be able to use it and continually review its accuracy and usefulness. Commercial platforms will often come with comprehensive reporting tools from basic dashboards with KPIs like D2/D7 Retention, DAU (Daily Active Users), ARPU (Average Revenue Per User) and even ARPDAU (Average Revenue Per Daily Active User). There are two essential tools every developer should look to use. First there are 'Funnels': we set up events in order of how

a player passes through the game we can see where in the pipeline you have the most problems. The second are heat maps which put that data into your level in a way which lets you understand exactly whats going on inside the gameplay session.

Free vs. Paid (and Subscription)

The arguments between free and paid games have become almost tribal amongst game developers asking whether business models have tarnished the nature of game design, even asking questions about the morality of these money focused designs.

Looking in economics terms, it is clear what happens when supply goes up, prices fall. With an effectively infinite supply, the price falls to zero. This is exactly what has happened and why Free2Play is so dominant. But wait, what about the 7% of revenue for iOS on premium games? Successful premium games are the ones which have been able to attract an audience by offering something they perceive to be of greater value than the rest of the games available. Games like Monument Valley or The Room have shown that this is still possible, and they are noticed because of their premium pricing. This has not been at the scale of the revenues of the top performing Free2Play games, despite considerable profiling by app stores.

The easiest way to understand Free2Play is to realize that these games have simply taken the retail side inside the game. That means that the people who know the game and its players best (i.e. the developers) can identify items that players will love that compliment and enhance the experience and sell them directly. The movement towards free has not proved an easy path for many game providers and attempts to 'clone' the business models of games like Clash of Clans or Candy Crush have rarely seen even a comparable level of success.

This is despite the formula appearing on the surface to be so simple and can quickly become 'not much fun' causing a lot of players to churn. If your player feels that the game is merely an exercise in getting me to open my wallet, just how engaged will I be as a player?

Advertising is playing an every increasing role in games and it is fascinating that unlike other media, ads in games do not seem to cannibalize the audience. Similar to the way we are changing how we look at in-app purchases, we are seeing a movement looking to find way for ads to add value to the overall playing experience, rather than just blocking the players progress. The use of banner ads in a game, which can be clicked accidentally and take up valuable screen space, is largely being replaced by interstitials or opt-in video. We are also starting to see the introduction of brand based ads which, perhaps surprisingly, seem to add a level of credibility to the whole experience; apparently seeing an ad from Coke or Audi makes players gene more engaged.

With the arrival of streaming and subscription services a range of new opportunities for developers of certain types of games. At the time of writing Apple Arcade is focused on exclusive 'paid' style games being added to a library for subscribers to enjoy as an all-you-can eat model - very much the same model as a Netflix. This is likely to be an extremely interesting business model for many developers who are quick to the release or who are focused on very high quality single-play through experience titles but it will not suit everyone. Similarly Stadia and a number of retro-game streaming sites are being announced using a similar model; but there are different requirements in terms of maintaining value for the player over time. The risk of this model for a developer is similar to that of music artists on Spotify; the more games supported the more the revenue is shared out and the more it

gets focused on who is currently promoted. "All-You-Can-Eat" subscription models as much friction as "Paid upfront" games even if you have a free trial period; and although they can deliver on-going revenue they lock the games growth to the audience - you cannot offer them more things that increase revenue per user only retention. "Battlepass" style methods as seen in Fortnite have shown more player friendly approaches where a clear value offer is made and players can see what they are missing out on. But importantly you can still up-sell new things to those players especially through events and updates. Battle-pass is often opt-in rather than opt-out at the moment but will increasingly include subscription options. Over the next few years we will see smarter use of subscription models to increase engagement and to complement other forms of revenue including Ads and IAP on top of the subscription - but only where players get extraordinary value. And not all games will benefit from the model.

Seven Rules of Monetisation Design for Games

Rule 1: Utility

Everything starts with utility, an economics term in this case used to express the 'expectation of value' for the player. In Free2Play players are usually not buying the 'gems' or what ever currency you might be using because they are so shiny. Buyers are driven by the expectation of what gameplay these items will unlock. The same principle applies even if you are charging for access to the game. It is not the physical delivery or the download size that players value but the anticipation of the game.

While in premium games we earn money before the users are actually experiencing it (apart from potential additional DownLoadable Content (DLC)), for free2play games the lifecycle of the player after the initial download plays a critical part in monetization. With this in mind it is often helpful to break your game into three sections:

— **Mechanic:** The core element of play - usually a loop usually with some kind of challenge; resolution and reward
— **Context:** A mechanic used to drive a sense of purpose and progression including narrative, unlock tree, boss modes, etc.
— **Metagame:** The elements that are not about the pixels including collaboration, clan systems, mode of use of the devices, etc.

Each of these stages may include an opportunity for the use of an in-game advert or in-app-purchase (IAP).

The levers we use to add value will usually be different in these different stages of the experience. Consumables which work as a part of the mechanic, might be inappropriate for the context or metagame. Splitting the experience down into sections allows us to understand where the value really is and keeps the focus on the benefits for the players. We can also use this mindset to rethink how we use ads especially opt-in video ads. They can now become a method to extend players behavior, e.g. watch an ad to get a free power-up you have not tried before. Once the player tries it hopefully they will be more inclined to spend a little money on a bundle of 10. Each purchase decision has a different impact depending on where in the game it occurs as well as the players lifecycle. The use of analytics and even machine learning can transform our success as long as we set-up meaningful tests.

To be more specific about what monetization elements we want to use in our game design it is worth looking at it through the lens of our game design toolkit.

Start by looking at the types of goods in games:

— **Sustenance:** What are the elements we require to continue playing?
— **Shortcuts:** What factors increase our chance of success or reduce the impact of failure?
— **Socialisation:** How can players express themselves and their progress in the game?
— **Strategy:** Can we introduce new playing options (without breaking the game)?

Those goods come in different forms as well:

— **Consumable:** A one-time use item.
— **Capacity:** Something which limits growth/play
— **Permanent:** A permanent upgrade or unlock item
— **Generators:** An increase in the supply of a consumable

When we combine the type and form this creates a powerful way to assess your game's monetization model.

GOOD/TYPE	Consumable	Capacity	Generator	Aspiration
Sustenance	Fuel	Fuel Tank	Gas Station	Better Car
Shortcut	Strength Potion	More Strength	Alchemist	Improved Recipe
Social	Heart Gift	Supporters	Wanderers	New Outfit
Strategy	Door Key	New Tool	Booster Pack	Red Katana

Let me illustrate this with a concrete example: We have a driving game where we use fuel as an energy mechanic; but we may decide that you cannot buy fuel (we do not want it to feel like a tax) but we do allow players to watch an opt-in ad to refuel once per 24 minutes. Players can also upgrade the size of their fuel tank which means they get more plays before running out. Then we can have a gas station which means they can replace their fuel faster than before. Finally we have the better car which we desire because it adds advantages; but it also has consequences e.g. using more fuel making this a playful choice - not just an upgrade.

Rule 2: Anticipation
An essential element of monetization design is how we communicate the Utility we are creating to the player. There are generally four forces preventing a player making a purchase: Uncertainty of outcome, Social issues, Opportunity costs and External needs.

A hard lesson from this is that we cannot make people pay; and although it might be possible to manipulate people in the short term that is not sustainable, counterproductive and damages trust for everyone. Instead, we need to create the conditions where people can give themselves permission to play. This means we need to create the following four factors: Expectation of delight, Social capital, Call to action and Abnegation of other priorities.

Rule 3: Scarcity
Just like in the real-life economy, scarcity is also a vital aspect of the economy of your game. However, when creating scarcity, do not forget about rule1: Utility. Any use of scarcity has to be authentic and focus on the enjoyment of the game. Adding opt-in video ads or IAP must be an extension of our overall

design and we must consider how spending affects the balance of the game to avoid creating a 'Pay To Win' model (better described as a broken game).

Vital to scarcity is to look at the opportunities in play and through pay to earn those scarce resources and currency and to make sure that you have designed appropriate 'resource sinks' which ensure are not directly about 'winning'. We need soft variables of success to deliver the maximum Utility to the player.

Look at the game Rock-Paper-Scissors. Imagine we have two upgrades 'lizard' and 'spock'. Because these new shapes can both win and lose against the others the actual chance of random success is unchanged. There is no technical advantage. However, there is a psychological advantage as well as increasing complexity limiting peoples ability to second guess their opponent.
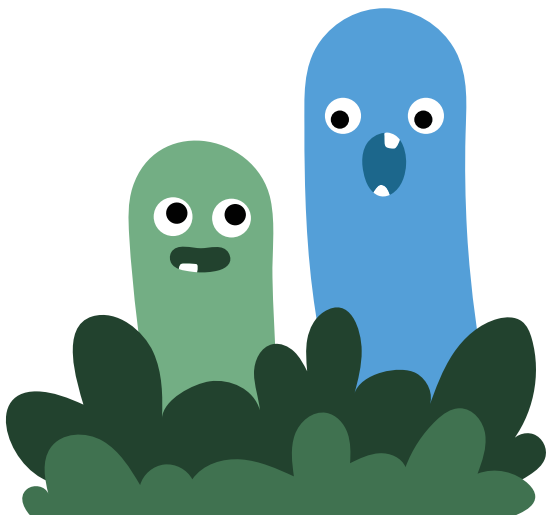
Rule 4: Timing

Players needs are not static. This is especially important in free2play games but also affects players willingness to make DLC purchases in premium games. As developers we have to think about the player lifecycle and how that impacts players willingness and interest in making purchases. It is also important that the game feels alive through community engagement, events and regular predictable updates.
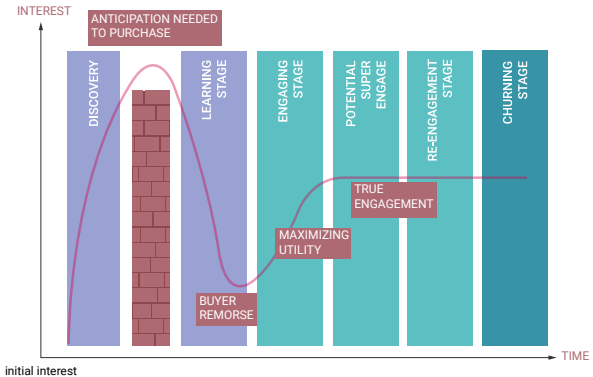
- **DISCOVERY:** Will the player install and play
- **LEARNING:** Will the player learn the game and make it part of his/her routine
- **ENGAGING:** The player is committed to the game and potentially willing to spend
- **POTENTIAL SUPER-ENGAGE:** Is this a player willing to invest more in this game than average
- **REENGAGEMENT:** Player is at risk of churning; can we bring him/her back
- **CHURN:** Player has moved on

If we understand the status of the player we can be in a better position to understand what they value and how we can make offers which extend their lifecycle and their life-time value.

Rule 5: Repetition

Repetitive actions can become intrinsically rewarding and this can also help build positive habits as well as building trust with your players that they can obtain the Utility they anticipate from your game. Highly repeatable game mechanics are essential to monetization success, especially on mobile where we can play repeatedly throughout the day.

This is reflected in the rapid increase in the willingness of player to spend more the longer they have engaged with a game.

In 2014 Unity Ads did a survey with 3000 online participants which showed a clear correlation between longer play time and increase revenue. This was not just linear either.

HOURS PLAYED	AVERAGE SPEND
10+h	$15.15
5-10h	$4.90
1-4h	$2.55
<1h	$0.66
ALL	$4.58

However, it is also important to avoid player fatigue. This means giving players natural breaks where they can switch between 'Frustration' to 'Relief'; from 'Intensity' to 'Relaxation' and feel that they can break away from the game with a reason to return later. Otherwise they may burn out. This can play an important part in the placement and use of different

ad formats. Blocking ads like full screen interstitials can be frustrating so work best on screens where the player can rest. On the other hand opt-in ads are less frustrating and by their nature offer the player 'Utility' in return for engaging but this creates another design problem. We have to think about what benefit makes the most sense for the player to watch that ad - it is almost an extension of the thinking we need to do for IAP.

The way we pay (or exchange time) for gameplay can have different impacts on player fatigue and there is a spectrum from the least impact i.e. In-Game-World Brand Ads (paid by views), the least impactful, to Rewarded Video, In-Game Currency, Interstitials, Paid IAPs and finally, the initial purchase of a Subscription.

One of the keys to sustaining interest over long periods of play lies in the use of predictable uncertainty. Take 22 Cans game, The Trail. In this game, items we may need for crafting appear along the trail at apparently random intervals, we can only carry so much in our bag and each time we take the walk we have to decide if we will pick up those items we find. If we do we risk not having enough space for a specific item we may need to progress. It is predictable we will get those items; but uncertain when so the player is entertained by the time they spend looking whether they appear or not. The act of looking does not become boring even though it is repetitive because it retains a sense of unfulfilled anticipation.

Hypercasual games appear to break the rules on frustration, however, this is because there already is a low threshold of engagement; showing ads arguably becomes a relief point from the intense repeating gameplay.

Rule 6: Evidence

We have talked about the importance of data. Here is a set of practical steps for implementing analytics.

1. **Define the objective** Ask why you want this data and set clear objectives on what you expect the data to tell you
2. **Identify what you can measure** Identify what data points will help us answer the questions laid out in our objectives
3. **Identify player decision points** Identify the trigger actions in the game which indicate player decisions
4. **Define common events** What do not we have to specifically collect and which data point allow us to compare events across the game e.g. AnonPlayerID; SessionID/etc
5. **Identify reference data** I.e. What do not we have to specifically collect because these data points do not change during a player session (i.e. we need not repeatedly capture them)
6. **Select your analytics platform** Do you make you own or use an external provider such as Unity Analytics
7. **Segment your data** Create custom cohorts so you can compare different parts of your audience or different builds of your game, etc.
8. **Create funnels, heat maps, define KPIs** as required to track your games performance
9. **Continue to iterate and test** your reporting process as well as your game

Rule 7: Scale

The most troublesome rule; and perhaps the one rule to rule them all. Scale matters. We need an enormous volume of players watching a significant number of ads per day in order to generate measurable revenues just on ads. Typically 30%-50%

of players choosing to watch an opt-in ad. IAP are much rarer, but their value is more tangible. Typical games see 2-3% of players spending on IAP of the whole base which downloaded the game. However, this statistic really misses the point. The percentage of people who spend the most money will be those who remain playing your game longest. At 30 days we expect to see less than 10% of the original downloads. If the average spend on any item in the games is as little as $1 this also depends on scale.

Scale is not just about the number of players. Puzzles and Dragons and Clash Royale do not have the same size of audience as Candy Crush but they do see enormous revenues.

The key to scale is getting more players, doing more things, more often for longer.

This is an industry led by acquisition, retention and monetization; but we can be smarter if we apply utility, anticipation, scarcity, timing, repetition, evidence and scale.

Getting Discovered

If you have followed these guidelines then you will have already put your game design into the best form that suits your audience and that itself will (hopefully) give you a fighting chance. However, that alone is not enough. We have to use every possible communication route we can and that usually requires investment. It is still possible to succeed without spending money on advertising, but you have to be the winner of a global lottery ticket. This applies on mobile games as on any other kind of mobile app as well. Hints how to market your software can be found in the monetization and in the ASO chapter. Additionally, here are strategies which you might want to think about especially for games.

Press

Getting noticed by the press can help, particularly if you participate in games awards such as Pocket Gamer's Big Indie Pitch[7] or the Indie Awards at Casual Connect[8]. If you can get the attention of YouTubers that may also help. However, the benefit of these activities is rarely measured in downloads. Instead they give you more credibility especially with the App Stores.

Advertising

Spending money on advertising can help, but it is important to realize that you are competing with a lot of people and some big players who are seeking large audiences. It is important to remember what you are trying to achieve when creating an advert. There are two motivations, building awareness and direct action (i.e. downloading the game). In games we are able to put adverts in other games and apps on the same device we want the players to experience the game. There is nothing getting in the way between the advert and the app store. One click and you can buy/download the game. That is an amazing thing, no other media has that kind of frictionless experience.

Another peculiarity to be aware of is that the larger the reach (range of players) you are looking for, the more expensive each of the installs. This is because buying space on an advertising network is based on a bidding process and the results will be calculated on the basis of Cost Per Install, Cost Per Mille (i.e. per thousand) or a blend of the two known as eCPM (effective CPM) as well as ad networks like *Chartboost.com* or *AppFlood.com* which offer cross promotion.

---

[7]   pocketgamer.biz/events

[8]   indieprize.org

Video based advertising is growing and allows the player to instantly understand the nature of the game being shown. This is often combined, such as with Unity ads[9], *Vungle.com* and *AdColony.com* with incentives inside the game – such as free currency or power-ups. This kind of in-game reward is different from external off-platform offers that can be restricted on Apple's platform.

In Game Events

Regular events and outreach to the community allow us to sustain and to grow our audience. Building on genuine social experiences, such as the recording of gameplay videos and sharing of community data (high scores etc) players can help reach out to their friends and other potential players via Facebook, Twitter, Everyplay and YouTube.

Influencer Marketing

In recent years Influencer marketing has become increasingly professionalized and working with YouTube personalities continues to have a significant impact on the take up of games, including mobile. Increasingly, even moderately successful influencers will have agents who not only protect their interests but also help developers match up to the best people for their game brand. It remains vital, even in the game design process to consider how your player will look good to their audience whilst playing your game. Not ever player is a YouTuber but thinking about the visual impact of the experience in this way will make it easier to not only entertain your players but also to build up awareness for your game brand. Any engagement with a Youtuber needs to be done carefully as poor behavior

---

9   unityads.unity3d.com

on their part not only damages their own reputation but can affect your own - even if only in the short term.

eSports

In 2018 sports captured the attention of nearly 400 million viewers worldwide — and cable and OTT platforms took note, with media rights revenues topping $180 million.

Total esports revenues reached $869 million in 2018, and is forecast to more than triple by 2022, reaching $2.96 billion, according to an October 2018 report by Goldman Sachs.

The level of talent and professionalism in the eSports market is now significant and game developers are starting to consider how this will impact game design. However it is still the case that there are very few mobile games which can legitimately claim to have gained a strong enough following.

In Summary

In the end despite all the differences in the details, mobile is like any other platform. We have to acquire, retain and monetize our audience. That only happens if we entertain players in the way that works for their devices. Devices which are perhaps the most social and most pervasive devices in human history. Mobile gaming is thriving despite the hurdles and the lessons learned will affect every aspect of game development.

BY Dean Churchill

Security & Privacy

The mobile app galaxy has grown way beyond the constellations of iOS, Android and a few other smartphone platforms. Applications are being developed and deployed on an ever-increasing panoply of devices: game consoles, TV platforms, virtual/augmented reality, drones, wearables, screen mirroring devices, in-car infotainment screens. These apps are connecting to cloud services, and managing and interacting with IoT devices. The application space runs from the most innocuous flashlight app to critical apps for monitoring the location, safety and well-being of people with chronic health conditions. Beyond business, finance and social networking, mobile apps are controlling home security and surveillance systems, operating drones and running industrial tools. As the value chain of mobile apps is increasing, so are the unrelenting efforts of hackers to compromise these software systems to steal money, throw off elections, spy on adversaries, stalk, commit fraud, and take down or take over networks. At the same time, regulators around the world are writing new laws regarding the privacy of data that have celestial reach. It is a double-chase drama: you risk being pursued by the bad guys and the good guys at the same time. Criminals want to steal or ransom your data, while regulators will fine you if you fail to secure and protect your customers' data.

So now more than ever, you need to focus on designing secure end-to-end solutions utilizing your mobile app as an end point, and providing privacy protections for the data the app uses. This requires thinking of the mobile app in the context of cloud and backend services, and knowing where and how your data is stored.

This very brief guide to security and privacy discusses a couple approaches to threat modeling, key issues in privacy regulations, and an approach to security testing. While the task of securing an app may seem daunting, the majority of attacks on applications, be they web, desktop or mobile apps, can be averted by simple secure-coding techniques and basic security testing.

Threat Modeling

The starting point for secure IT designs is to create a threat model. A threat model begins with listing all the assets of concern to your product. Assets include all the software components, hardware, data schemas and data stores, APIs, microservices, interfaces, and communication channels needed to make your product work. Then create a model of threats for each of these assets. There are multiple threat models available that help with the analysis.

The OWASP Mobile Top 10[1] list identified the dominant threats to mobile platforms: improper platform usage, insecure data storage, insecure communications, insecure authentication, insufficient cryptography, insecure authorization, client code quality, code tampering, reverse engineering and extraneous functionality. OWASP provides extensive supplementary materials on the nature of the threats, and how to code securely against them, and test the applications.

The Cloud Security Alliance (CSA)[2] is focused on threats related to cloud services, and provides training and tools for assessing threats to cloud computing. CSA offers a free Cloud

---

1 owasp.org/index.php/Mobile_Top_10_2016-Top_10
2 cloudsecurityalliance.org

Controls Matrix that aids in assessing the risks of a cloud provider.

The STRIDE security model[3] focuses attention on software design, to protect against specific threats of **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**scalation of privilege. This model applies equally well to server-side software as well as mobile clients.

Security controls are those pieces of software written and practices followed that reduce the risks to all the assets. For example, each of the threats enumerated in the STRIDE model can be countered by one or more design elements of your application.

Spoofing

Spoofing attacks have two basic types. The identity of a user might be spoofed by a hacker, pretending to be someone he is not. This is where strong authentication comes into play, to be sure that users of your app are who they actually claim to be.

Users may also be victims of spoofed services. The classic phishing attack is where you receive a spoofed message (SMS, email, voice call) asking you to log into your bank account at the given URL. Often the message indicates a sense of urgency. But the URL is not for your expected bank, but a link to a web site that looks just like the bank. The hackers are hoping someone will enter their username and password there, then they can use those credentials to log into the real bank and transfer money out.

While you cannot stop hackers from sending spoofed messages to your customers, you can have a policy of not contacting your customers to ask them to log in. Tell your customers that you will never call them asking for their password, or send

3   en.wikipedia.org/wiki/STRIDE_(security)

unsolicited mail with URLs to your web domain. You can ask them, instead, to download your mobile app from a public app store, and log into your service via the app. That process is harder to spoof, as long as there is no malicious app in the app store that looks like yours. To that end, you need to keep an eye on the app stores and make sure your company's name and intellectual property are not being used by any apps you have not authorized.

Tampering

Tampering attacks may be directed against software, data, databases, files, and communications networks. Tampering of software packages is prevented by digitally signing packages (like iOS IPA and Android APK files). But once installed on mobile devices, hackers with rooted/jailbroken phones can manipulate the software and data at will. They can replace libraries with hacked libraries, change the contents of memory at run time, patch the binary, and otherwise subvert the application for their own benefit.

Hence your design must not trust the user's device. Countermeasures include: do not store secrets (passwords, API keys, sensitive data) in your IPA or APK files, as hackers have full access to all the information therein. Inspect the Android Manifest file in APKs, and the `Info.plist` file in IPA files to be sure no secrets are there. Generally any kind of sensitive computation, like user identification, authentication and authorization, or a proprietary algorithm, should be performed server-side to keep those functions out of view of hackers. Sensitive data should not be stored on the mobile device if possible, otherwise use an encrypted storage mechanism, like the keychain available for iOS and Android. Tampering of data in storage can be detected using secure cryptographic hashes and digital signatures.

Many mobile apps are using a database that runs on the device, set up by the application. Not only can hackers view the contents of these databases, but they can also modify the data and the schema to suit their purposes. So do not store sensitive data in the database on the device. The mobile client needs to validate that fields and rows of data retrieved from any database (client or server-side) meet the expected type, size, quantity, and range of values.

Tampering of networks, exploited through man-in-the-middle attacks, can be prevented by the use of HTTPS, certificate pinning, end-to-end encryption and use of VPNs.

Do not use self-signed certificates in your applications, except perhaps in a development environment.

Repudiation

Repudiation is countered by secure logging of user identification and users' actions, which in turn requires solid user identification, authentication and authorization functions. Use of 2-factor authentication is increasingly common, and this helps to control access to applications.

Many applications and services use the mobile device as a second factor for authentication. The service sends an SMS message to the phone containing, for example, a one-time 6-digit token, and the user enters that token in the application to confirm the user's identity. But in the mobile app space, if your phone has been stolen, and a hacker knows your password to get into a mobile app, having an SMS message sent to the stolen phone provides no additional protection at all. And of course, social engineering hackers are focusing on that second factor as well.

Information Disclosure

Information disclosure related to data is countered by encryption in transit and in storage. Be certain to use current, industry-standard encryption standards. All current mobile devices support encryption with AES, and with keys sizes of 128 and 256 bits. But then the encryption keys need to be protected carefully, as a leaked key will allow hackers to decrypt the data.

iOS has a key store for protecting sensitive data. Android has the Keychain API. These key stores will store data encrypted on the device, making it difficult for hackers, even with root access to the device, to steal that information.

The software itself is easily disclosed from a mobile app. Android apps, in particular, written in Kotlin or Java, are readily reverse-engineered to reveal their underlying source code. Code obfuscation tools can be used to make the code more difficult to decipher.

Denial of Service

Denial of Service attacks are usually seen on the server-side. Botnets are a continuing problem, as they attack servers, and deny access by legitimate mobile apps. One strategy for reducing the risk of DOS attacks is to use a commercial cloud service. Cloud services all have DOS protection built into their infrastructure, so that you can focus more on the business functionality of your applications.

Escalation of Privilege

Escalation of Privilege may happen in the model context where network traffic may be hacked, and privilege level of a user may be elevated, to access data the user should not see. This is more of an API hack than a mobile app hack per se. But the risk is still there. You counter the risk of Escalation of Privilege

by having well-defined roles that are enforced by access controls: Minimize privileges – the fewer roles and privileges the less "attack surface" the application has.

Some mobile apps are nothing more than an icon that launches a web site. If your solution uses web services and web sites, then the OWASP Web App Top 10[4] is a standard list of keys issues to be concerned about.

Security Testing

Once you have built your application, and completed all the functional testing you can[5], it is time to perform a security test. In the mobile app arena, you can hand off the binary file (IPA or APK) to a separate security test team, or do the testing yourself.

Here is a checklist of security tests you can run your app against. This is by no means comprehensive, but gives you an idea of the scope of issues to examine once you build an app and are ready to test it for security:

1. Identification – does your app identify the user of the app appropriately? Some apps, like a flashlight app, are designed for use by anonymous users; others, like banking apps, need to identify each user uniquely.
2. Authentication – if the user needs to be identified, do you have a secure way to authenticate, to prove the user is who he claims to be? Make sure that hackers cannot bypass the authentication process. Authentication needs to take place on the server-side – in the cloud, in a datacenter, anywhere but on the mobile device itself, as

---

4 owasp.org/index.php/Category:OWASP_Top_Ten_Project
5 See "Testing" chapter in this guide

the mobile device cannot be trusted. Consider the use of 2-factor authentication. We are well past the era where systems can be secured reliably with just a password. Anyone who is logging into an app while in public is at risk of a shoulder-surf attack - where someone makes a video of you entering the password in the app. And phishing attacks are still ramping up - emails, SMS, other messages telling you to click on this link and log into your bank account before it gets closed - or some other ploy that appeals to emotion, urgency, or threats to get you to log into a fake service with your valid password and id. Second-factor authentication makes it more difficult for a hacker to get into your accounts with just a stolen password. But if that second factor is just an SMS to your stolen phone, there is no added protection.

3. Authorization – once the user is appropriately identified and authenticated, does he have the authorization to use whatever service you are providing? If it is a paid service, you need to test that hackers cannot find a way to use your service for free. Authorization functions also need to be run on the server-side.

4. Accountability – if users are thus identified, authenticated, and authorized, does the app log import transactions, so that no repudiation attacks can occur? You want enough log information, without compromising the users' privacy, so that if a government auditor comes soliciting information on who used your app, when and for what purpose, you can tell them. And if the user comes and demands "forget me" you can account for all the data collected pertinent to that user and securely delete that data.

5. App code and web security – does the app resist injection attacks of all types? Protect against injection attacks by sanitizing input from untrusted domains (consult your threat model here). Validate all input for proper size, type, range of values, and quantity of data expected. Run secure static code analyzers to find subtle security issues in your code, and remediate them. Two leading vendors for secure static source code scanning are Checkmarx[6] and Microfocus Fortify[7]. Are the sessions secure, so that APIs cannot be exploited without proper authentication? You may need to use web a pen test tool, such as HCL Appscan[8] or Fortify WebInspect open source software[9] to make certain your backend services are secure. Use code obfuscation tools, such as ProGuard[10] to make an analysis of your application more difficult.

6. Packaging security – check to make certain that you did not include in the app package any sensitive information, such a test environment information, passwords, developer documentation. If using third party libraries, check that you are using the most recent, patched, and reliable versions of those packages.

[6]   checkmarx.com
[7]   microfocus.com/en-us/products/static-code-analysis-sast
[8]   hcltechsw.com/wps/portal/products/appscan
[9]   microfocus.com/en-us/products/webinspect-dynamic-analysis-dast
[10] guardsquare.com/en/products/proguard

7. Data security – if you store any data locally on the data, realize that it can be viewed or changed by someone with a rooted or jailbroken phone. Even if encrypted, the encryption key, if stored on the device, will enable a hacker to steal that information. So check that you have not stored any sensitive information in databases or files on the device.

8. Transport security – when running the app, use Wireshark[11] or a web proxy tool, like Paros[12] or Charles Proxy[13], to capture traffic going to and from the app, to be sure that all sensitive information is transmitted encrypted, and only to the expected servers. Even non-sensitive information needs to be encrypted to assure the integrity of the data, lest a hacker injects malicious content into the network traffic. Do not support any TLS protocol older than TLS 1.2 if you can avoid it. The TLS 1.3 protocol[14] was approved as a standard by IETF, August 2018, which improves on the security and performance of TLS 1.2. The Qualsys SSL Server Test[15] is a free online service that will perform security testing of the TLS certificate for any public domain name on the Internet. It will assure that

11 wireshark.org
12 sourceforge.net/projects/paros
13 charlesproxy.com
14 ietf.org/blog/tls13
15 ssllabs.com/ssltest
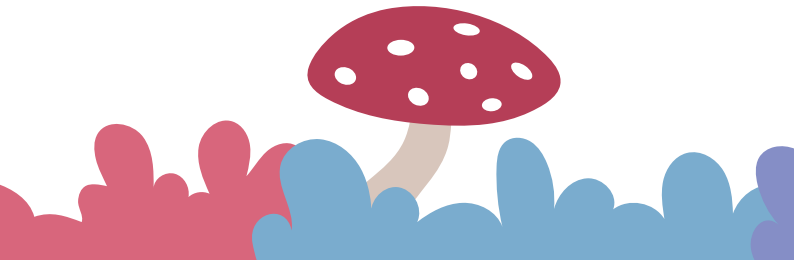
the TLS certificate is not expired, has a trusted certificate chain, will flag and identify known security exploits, show which cipher suites are secure and which are weak, and compare the domain name against the common name on the certificate. Use this to test all the HTTPS domains that your app uses, especially ones related to 3rd-party software packages -- like services for ads, metrics, social media, maps, etc. On iOS apps, use the App Transport Security settings to force the use of HTTPS if possible.

9. Protocol security – make certain that login and logout protocols work correctly and cannot be bypassed. Logout functions need to send transactions to the server to close any secure sessions – do not just revert to the login page of the app. Check that password reset functions cannot be spoofed easily. Set a password policy that requires complex passwords.

10. Privacy – make certain you know what data is being collected about your users. Minimize that data to reduce risk. Do you provide the user with a privacy notice? Did you build in "forget me" capabilities to meet GDPR privacy requirements?

Meeting Privacy Requirements

Most of the above are focused on locking down your application, so that information does not leak, via passive surveillance or active intrusion. But current privacy regulations, like GDPR, requires the ability "to be forgotten", to securely destroy a user's data in the system, secure the right of a user to access his own data, and to be notified promptly of a breach. And this must all be "watertight". Most products are designed to collect information, and usually do not have a secure way to remove data from the system, short of relying on a database admin and support ticket to try and remove someone's data. The penalties for non-compliance can be staggering, the reach is global - anywhere the data of an EU citizen is being processed GDPR applies. Consent has to be explained in clear and plain language, and must be as easy to withdraw consent as to give it. So mobile apps not only need to be technically excellent from a security perspective, but they also need to implement effective data management tools – server and/or mobile side – to comply with GDPR the sun within a solar system of privacy laws and regulations.

The State of California has its own privacy rules, the California Consumer Privacy Act[16], that takes effect on January 1, 2020. Other U.S. states are considering their own privacy laws. Make sure you stay on top of them.

[16] en.wikipedia.org/wiki/California_Consumer_Privacy_Act

Learn More

Here are some useful resources and references which may help you:

— Apple provides a general guide to software security[17]. It also includes several links to more detailed topics for their platform.
— Commercial training courses are available for iOS and Android. Lancelot Institute[18] provides secure coding courses covering iOS and Android.
— A free SSL tester is provided by Qualsys Labs[19].
— Extensive free application security guidance and testing tools are provided by OWASP[20], including the OWASP Mobile Security Project[21].
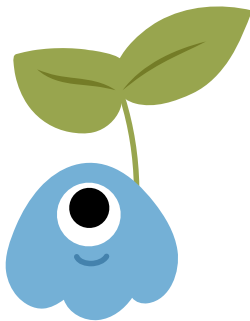
17 developer.apple.com/library/mac/navigation/#section=Topics&topic=Security
18 lancelotinstitute.com
19 ssllabs.com/ssltest
20 owasp.org
21 owasp.org/index.php/OWASP_Mobile_Security_Project

The Bottom Line

The galaxy of mobile apps is becoming ever more intertwined with the expanding universe of cloud services, IoT devices, and exotic new platforms. More than ever these apps are exposed to those who would like to use the apps maliciously, steal data from them, hold the data for ransom, or attack connected services and devices. The appropriate level of application security is something that needs to be considered on an end-to-end basis. In the end, your app and its backend infrastructure and endpoints will be in-the-wild on its own and will need to defend itself against hackers and other malicious threats.

Invest the time to learn about the security features and capabilities of the mobile platforms you want to target. Use techniques such as threat modeling to identify potential threats relevant to your application. Perform code reviews and strip out non-essential logging and debugging methods. Run a secure code analysis tool against your mobile code to find vulnerabilities. Consider how a hacker would analyze your code, then use similar techniques, in a safe and secure environment, against your application to discover vulnerabilities and mitigate these vulnerabilities before releasing your application.

Pick trustworthy services and partners. Often we use and rely on third-parties to serve our users and stakeholders. Their approaches to privacy matter as do their practices. Using companies who hold themselves to high standards and who are clear about what they do - and do not do - can help us protect our users and stakeholders.

And take the time to consider the privacy implications for any data you collect about or on behalf of the user, particularly if storing or synchronizing that data into the cloud.

BY Cathy Rundle

Accessibility

Why Accessibility is Important

According to the World Health Organization (WHO) over 15% of the world's population has some form of disability[1] and rates of disability are increasing due to population aging and increases in chronic health conditions, among other causes. This means that around 1 billion potential users could have difficulties using your app if your app is not accessible.

There has been a huge increase in smartphone and tablet use in the general population, this is no different for those with disabilities. The WebAIM Screen Reader Survey[2] shows that there has been an astounding increase in smartphone use by blind people who use screen readers. Older people might not have used a computer at work; however, they are finding that they can get to grips with touch screen devices more quickly than a traditional keyboard and mouse. As our population ages, the levels of disability increase and this means more and more people will have difficulty accessing services in traditional ways. Providing an alternative accessible digital solution will ensure disabled people can continue to be independent.

For example, if they are unable to get out of the house to do their shopping or banking, then providing accessible online services means they can access these services independently. It is important to recognize how important independent access to services is for people with disabilities. Apple has put some videos together with disabled people showing how they use

---

1   who.int/mediacentre/factsheets/fs352/en
2   webaim.org/projects/screenreadersurvey6/

technology in their everyday life[3]. This demonstrates just how important accessibility is to enabling independence.

There are lots of other reasons to make your apps accessible:

— Implementing accessibility can often improve overall usability: For example, if you ensure that every button and form element has an appropriate label, that is helpful to everyone, not just those with disabilities as the user will know how to interact with it. Making sure that the wording is clear and unambiguous is helpful for people with cognitive impairments and also for everyone else.
— It just makes good business sense: For example, people with disabilities have spending power and if they find an accessible app that works for them they will not only use it, they will also tell others. You may discover a significant new market when you develop apps that suit these users.
— Access to goods and services for all is the law in many countries: For example in the UK the Equality Act 2010 requires there to be access to goods and services for everyone and this does include services which are provided via electronic means such as websites and apps. Public bodies also have an anticipatory duty to ensure their services are accessible, so they cannot consider accessibility as an afterthought.
— Where accessible solutions are mandated by legislation, your app may be the only option for that business to realistically use: For example, your app may be able to tap into government-funded market sectors such as education where legislation, such as Section 508 of the Rehabilitation Act in the US, may mandate an accessible solution.

[3] apple.com/accessibility/stories

— The organization that the app is being developed for may
   have a corporate social responsibility (CSR) statement or
   program: For example, web and app accessibility provides
   social inclusion for people with disabilities which is a
   primary aspect of corporate social responsibility.
— Mobile platforms from Apple, Google and Microsoft
   leverage their accessibility APIs for UI automation testing:
   Making your app accessible can make automated testing
   easier.

What Accessibility Features?

Many of your potential users may have a disability that can
make it more difficult for them to use a mobile phone and
related apps. Disabilities include various levels of sight or
hearing impairment, cognitive disabilities or learning difficul-
ties, physical disabilities, dexterity issues, and so on.

   Many of these users rely on third-party software to assist
them in using their devices. This software is sometimes called
Assistive Technology and includes different utilities depending
on the type of disability. Traditionally, these types of software
or utilities have had to be 'added on' to a mainstream device,
often at a high cost, in order to make the device accessible
or easier to use for someone with a disability. However, many
smartphones and tablets now provide robust Assistive Technol-
ogy options built into the operating system that enable some
users with disabilities to use the devices without needing to
pay for extra Assistive Technology. What is offered depends on
the platform and the version of the OS. However - to work -
these features need your app to be designed well and support
these features.

— **Partially sighted users** - Someone who is partially sighted, benefits from being able to change the font size, font style, colors and use of bold and color contrast too. iOS, Android and Windows offer various options to change these in the settings. As well as the universal 'pinch to zoom' feature, iOS, Android and Windows offer a magnification or zoom feature, which enlarges a section of the screen and keeps this magnification level when moving throughout the phone or tablet. This has unique gestures associated with it and often each OS has its own unique gestures. iOS also has a built-in app that utilizes the camera on the phone to aid with spot reading on items such as clothing labels and restaurant menus.

— **Blind users** - Someone who is blind has to have information on the screen and navigation around the screen announced to them in synthetic speech. This is often called a 'screen reader'. iOS was the first OS to offer a screen reader built-in and it is called 'VoiceOver'. Android offers 'Talkback' which is fast catching up in popularity with the blind community as it is constantly improving. Windows first delivered the Narrator screen reader in Windows Phone 8.1 and it is even more improved now in Windows 10 Creators Update. Blind users may also make use of a Braille display, which is an item of hardware that provides feedback from the screen one text line at a time in the form of a line of Braille characters. Each Braille character consists of six or eight movable pins in a rectangular array. Most OS versions now support braille displays via Bluetooth.

— **Users with hearing loss** - Someone with a hearing impairment will often make use of a smartphone that is hearing aid compatible and offers features as iOS does such as 'LED Flash for Alerts' or 'Phone Noise Cancellation'. There

are also options in settings for iOS, Android and Windows to switch on subtitles and captioning. Making use of vibration for alerts is also helpful and haptic feedback has improved in recent versions of iOS in particular. A number of phones also provide support for hearing aids and teletype (TTY) devices[4].

— **Users with physical disabilities** - If a user has a motor impairment, they may well be using a third-party hardware product to access the phone, such as a switch as some devices do support this. Alternatively, they could be making use of voice recognition to access the device. Siri in iOS enables the user to access certain settings and functions and switch them on and off.

— **Users with a learning disability** - If a user has a cognitive impairment or learning difficulty, then depending on what the disability is, they may make use of the features in the settings that a partially sighted user does. Especially something like color options. Other users may make more use of voice recognition.

The overall experience for people with disabilities is affected by how well an app works with assistive technology. As these features are built into the OS and can be switched on in the settings, it is important that as a developer you recognize that they may be used with your app and ensure you test for this.

As screen readers and screen magnification programs in the OS make use of their own gestures, gestures in the app may be affected when screen readers or magnification are enabled. For

---

4   A TTY device allows people who have hearing loss or who are speech impaired to type messages to anyone else who has a TTY, using a telephone line.

example, a screen reader user can navigate a screen using left and right swipes or exploring the screen by moving their finger across the screen of the device in a consistent movement (explore by touch). As they undertake a swipe or encounter something underneath their finger, the item is announced. So an item is selected by touching once (which moves the focus to the icon) and opened by tapping twice. When using screen magnification, depending on the OS, the user may need to use a three-finger gesture. Including testing early on with accessibility features ensures that these gestures are supported by the app and that any redesign can happen before it impacts on users.

One of the best ways to learn more about these features is to switch them on and try them for yourself in different apps.

App Design Guidelines

The accessibility APIs look for text in specific attributes of standard UI elements. Screen readers used by blind people, such as VoiceOver and TalkBack, transform the text into synthetic speech which the user listens to. The screen reader software may also determine the type of control and related attributes to help provide the user with more contextual information, particularly if no text is available. It is important that the user understands what the label of the control is, what the control is and how to interact with it. In some instances, there may also be a tooltip to give extra information.

Just as web developers make use of standards and guidelines such as WCAG 2.0 to make accessible websites, it is important that as app developers, you do the same. At present, there is no de facto industry standard for app accessibility, although there are standards out there that can help.

The international standard, ISO 9241-171 ('The Ergonomics of Human-system Interaction: Guidance on Software Accessibility')[5] is a helpful standard as it is platform agnostic. This covers elements of accessibility and usability for a wide range of software.

The Royal National Institute of Blind People (RNIB)[6] have created a pan-disability app standard and testing process based on their experience in this area of accessibility. Their guidelines for native apps also reflects on principles from ISO 9241-171. They provide consultancy and training for organizations and agencies in this area and have an accreditation badge that can be awarded to apps that, following an audit

---

[5] iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.
     htm?csnumber=39080

[6] rnib-business.org.uk

process and user testing, are accessible. This accreditation is called 'RNIB Tried and Tested'[7].

The BBC has developed a set of BBC Mobile Accessibility Guidelines[8] that they use internally for their mobile content. Their guidance covers mobile websites, hybrid and native apps. They state that "they are intended as a standard for BBC employees and suppliers to follow however they can also be referenced by anyone involved in mobile development".

Here are some of the principles that are helpful to be aware of when developing an app. If you stick to them, you will also give your app the best chance of interoperating with assistive technology that the user may be running in conjunction with your software:

APIs and UI Guidance

— Find out what accessibility features and APIs your platform has and follow the best practice in leveraging those APIs if they exist.
— Use standard rather than custom UI elements where possible. This will ensure that if your platform has an accessibility infrastructure or acquires one in the future, your app is likely to be rendered accessibly to your users.
— Use the Accessibility API for your platform, if there is one. This will enable you to make custom UI elements more accessible and will mean less work on your part across your whole app.

---

7   For more information please contact RNIB Business at businesslink@rnib.org.uk

8   bbc.co.uk/guidelines/futuremedia/accessibility/mobile_access.shtml

— Follow the standard UI guidelines on your platform. This enhances consistency and may mean a more accessible design by default.
— The user should be able to apply their preference settings that the OS provides, such as accessibility settings.

Navigation

— Navigation should be logical and consistent. For example, if a back button is provided on each screen it should be located in the same place on every screen and consistently labeled.
— Support programmatic navigation of your UI. This will not only enable your apps to be used with an external keyboard but will enhance the accessibility of your app on platforms such as Android where navigation may be performed by a trackball or virtual d-pad.

User elements

— All user elements should be discoverable and operable via assistive technology, unless it is clear they are not required. Care must be taken when deciding that an object is not required and, for example, if the user touches a decorative image on the screen the focus must be taken to a sensible location either before or after the image and the user should not hear an error sound. Alternatively, an image within the text can be given alt text which will be spoken if the user touches the image.

— Where a user element has a status associated with it, that status should also be available to be read by assistive technology. For example if a toggle button is 'on' this should be announced by the screen reader. If the status changes, that should also be announced.
— Ensure touch screen targets are a reasonable size to ensure everyone can easily select them.

Labeling

— All elements, including form elements, buttons, icons and so on, should be labeled visually and programmatically with a short and descriptive name. The label should also be adjacent to the element it relates to.
— Each screen should have a unique descriptive name that relates to its content and aids navigation.

Colors and Fonts

— Ensure there is a good contrast between background and foreground colors. In particular, consider buttons which include text. Does the contrast between the text and the background color meet the ratio requirements in WCAG 2.1 or ISO 9241-171?

— Avoid using color as the only means of differentiating an action. A color-blind user will not be able to identify errors if they are asked to correct the fields which are highlighted in red for example.

— Consider the weight of the font, as well as color and size as a thin font, can be difficult to read

— Ensure that your app picks up the user settings relating to the font in each operating system. There could be times when this affects usability such as in a table when the layout may be lost, the font size for this specific area could be set to a maximum or preferably the layout adjusts appropriately for larger text sizes.

Notifications

— Error messages, notifications and alerts should be consistent, identifiable and clear. They should be announced by the screen reader and be clearly visible, ensuring they do not disappear from the screen after a short time period.
— Ensure that error messages, notifications and alerts are not provided by color/haptic/audible output alone. For example, someone with hearing loss will not recognize audible notifications.

Testing

— Do not forget to test your app on the target device with the assistive technology built-in to the OS with more than just the latest OS version. When testing on an Android device please remember that unless the user has a pure Android device, like a Google Pixel, they are unlikely to get access to all of the latest OS upgrades. This is because for OS upgrades you are at the mercy of your phone manufacturer, so the Android OS versions in the wild can be quite diverse. Because some handset manufacturers skin the OS, this can sometimes interfere with the way the accessibility features should work. Therefore, it is always recommended that testing for Android Accessibility is undertaken on a Google device. That way you can be sure there is nothing interfering with the way the accessibility features should work and you are working to a common denominator.
— Ensure your user testing includes people with disabilities too!

Apple, Google and Microsoft, have increased the importance of their respective Accessibility support by using the Accessibility interface to underpin their GUI test automation frameworks. This provides another incentive for developers to consider designing their apps to be more accessible.

Looking at the different mobile platforms more closely, it becomes obvious that they differ largely regarding their APIs, but they are starting to implement a lot of the same accessibility features.

Custom Controls and Elements

If you are using custom UI elements in your app, then, those platforms that have an Accessibility API enable you to make your custom controls accessible. You do this by exposing the control to assistive technology running on the device so that it can interrogate the properties of the control and render it accessibly.

You can get more information about Accessibility on Android from various resources on YouTube including the Google I/O presentations from 2017[9], 2016[10], 2015[11] and 2013[12].

The Apple developer program has helpful resources too. Take a look at their accessibility video presentations from the WWDC conferences available in the iOS Developer Center[13] by searching for 'accessibility'.

---

9   youtube.com/watch?v=h5rRNXzy1xo
10   youtube.com/watch?v=2qjgxH384Nc
11   youtube.com/watch?v=euEsfNR5Zw4
12   youtube.com/watch?v=ld7kZRpMGb8
13   developer.apple.com/wwdc/videos

Android App Accessibility

Accessibility was first a realistic proposition with Android 4.1 (Jellybean) and it is much improved since then. Since Android 7 (Nougat), there has been more prominent featuring of accessibility settings to let users independently configure their device. 'Display size' was added as an option and it gives the user an alternative view of the standard screen, making the icons and text larger as a native setting without any magnification or font adjustment. This is also available as a live preview.

Android 8 (Oreo) further enhanced accessibility by adding the possibility to control accessibility volume independently from media volume. It also introduced a shortcut, the accessibility menu, which can be added to the navigation bar allowing users to turn some accessibility features on and off from any screen. This brings up an accessibility menu on screen and users can customize the settings. Since Android 10, the accessibility volume provides quick access to volume and brightness controls as well and it allows users to enlarge the buttons.

After having been removed in Android 7, Google brought back the dark mode option (bright text on a dark background) with Android 9 (Pie). It allows the user to change colors or contrast using 'high contrast text' or 'color inversion' options. Since Android 10 the Dark Mode is available through the accessibility menu. This is equivalent to what Apple has undertaken with color filters in display accommodations. 'BrailleBack' works with Talkback for a combined speech and braille experience enabling the user to add a braille display via Bluetooth. There is also the addition of a 'Click after cursor stops moving' option, which helps people with dexterity issues or with low vision. Also popular is the addition of a large mouse cursor.

Accessibility features in the latest Android version include (but are not limited to) things such as:

Speech output:

— **Talkback** - Speech output for blind users. It includes TalkBack, Speak Select and Switch Access. Speak select can be set to read a selection or read the screen. This is useful for people who have some sight and find reading text difficult.
— **Enhanced search in TalkBack** - Provides a search to Talk-Back users allowing them to find text and move directly to that location on the page.

Display settings:

— **Font Size** - For partially sighted users and some users with learning difficulties.
— **Magnification gesture** - Zoom style magnification for partially sighted users.
— **Display Size** - For partially sighted users and some users with learning difficulties.
— **High Contrast Text** - For partially sighted users and some users with learning difficulties. This is an experimental setting.
— **Color inversion** - For partially sighted users and some users with learning disabilities who prefer an inverted color palette. This is still an experimental setting.
— **Color correction** - This can change the balance of how the color is presented on the screen and aids those with color blindness.
— **Remove animations** - Removes the transition effects.

Touch or switch settings:

— **Click after cursor stops moving or Dwell timing** - For those with dexterity issues or who are partially sighted when using a mouse connected to the phone, this setting allows users to set the mouse to automatically select when it stops moving.
— **Switch access** - For those with physical disabilities who prefer to access apps using a hardware device.
— **Touch and hold delay** - For users with motor control issues, this allows users to change the length of time before a touch becomes touch and hold.
— **Text editing for switch users** - Allows users to edit more quickly when using a switch device.

Audio or captions:

— **Captions** - Providing captions or subtitles for those with hearing loss.
— **Mono audio** - For those with hearing loss using headphones.
— **Live transcribe** - Provides captions in real time for presentations, video content etc.

Other useful settings:

— **Volume key shortcut** - Allows users to set a feature to be turned on and off by pressing both volume buttons at the same time.
— **Haptic settings** - Allows users to adjust the intensity of the haptic feedback or disable it.
— **Voice assist** - Overlays numbers to elements on screen allowing them to be selected using voice e.g. tap 7.

- **Accessibility timeouts** - Allows users to increase the timeouts in an app.

Google provides a good entry point to the topic in their developer portal[14]. You will also find more examples in the related training area in a section entitled Implementing Accessibility[15]. For specifics on how to use the Android accessibility API along with details of best practice in Android accessibility, please see Google's document entitled Making Applications Accessible[16]. Testing the Accessibility is also covered online[17].

Some more helpful resources can be found in the Support Library[18] which also includes ways to improve the accessibility of custom views. Finally, an interesting talk on Android accessibility from the I/O 2019 conference can be found on YouTube[19].

14 developer.android.com/guide/topics/ui/accessibility
15 developer.android.com/training/accessibility/index.html
16 developer.android.com/guide/topics/ui/accessibility/apps.html
17 developer.android.com/tools/testing/testing_accessibility.html
18 developer.android.com/tools/support-library/index.html
19 youtube.com/watch?v=Y_8ZlUP2C5o

iOS App Accessibility

Apple was the first company to embed accessibility features directly into the OS. Because of this, the support for accessibility in iOS is a little better than in Android, although Android is fast catching up. There are certainly comparable features now but it is a legacy issue as Apple was the first to move into this area. A lot of blind and partially sighted users also find the gestures in iOS easier to use and the experience more consistent.

With iOS 12 some of the accessibility features have moved from accessibility settings into display settings. You can now also use Siri to turn functions and features on and off on the device. This demonstrates that people are now recognizing that some of these settings are relevant for everyone. Accessibility is going mainstream as people just want to make the display more personal to them and have a device that is easy to use. This means it is even more important that developers consider accessibility settings when creating apps as the number of people using these features has increased.

There have been some hardware changes with the iPhones X and X+ as the home button has been removed. Haptic feedback is now natively deployed across the OS. If you use a 'picker wheel', it now delivers a discernible click when it is moving through options. This is a mainstream usability feature but certainly enables some disabled users to use the picker wheels more easily. Apple has also put more granularity into the haptic feedback that they provide, to allow the user to adjust the settings.

Some of the accessibility features in iOS include, but are not limited to:

Speech output:

— **VoiceOver** - A screen reader. It reads out the objects and text on screen, enabling your app to be used by people who are blind. The speech options also include speech select which can be set to read a selection or read the screen. This is useful for people who have some sight and find reading text difficult.

Speech input:

— **Siri** - This enables users to make phone calls and operate various other features of their phone by using voice commands. This can be helpful for a broad range of people including those with motor control issues, learning difficulties and vision loss.

Display settings:

— **Zoom** - This magnifies the entire content of the screen or a "zoom region" that can be used to magnify smaller areas.
— **Invert Colors** - This option inverts the colors on the display, which helps many people who need the contrast of black and white but find a white background emits too much light. This can be classic where all the colors are inverted or smart where icons and pictures retain their colors.
— **Color Filters** - This helps color-blind users differentiate colors and also aids those who have difficulty reading text on the display.
— **Reduce White Point** - This reduces the intensity of bright colors.

— **Larger Text and large accessibility sizes** - This can help a broad range of people from those who use glasses, through to partially sighted people and those with learning difficulties. Large accessibility sizes allow users to touch and hold on tabs or buttons and they will be shown in a large pop up in the middle of the screen.
— **Bold text** - This can help a broad range of people from those who use glasses, through to partially sighted people and those with learning difficulties.
— **Buttons shapes** - Removes the line (or shading on older OS) from button text.
— **Reduce transparency** - Reduces the transparency of backgrounds which makes the text easier to read for people with low vision.

Touch or switch settings:

— **Reduce motion** - Reduces the motion on user interface including the parallax effect of icons
— **Reachability** - Allows the user to bring the top part of the screen towards the bottom to make it easier to reach when using the phone with one hand.
— **Switch control** - For those with physical disabilities who wish to access the app using a third-party hardware device.
— **AssistiveTouch** - For users who have difficulty touching the screen or might need to create custom gestures.
— **Touch Accommodations** - These options give the user

the ability to change the settings, on how the screen will respond to touch gestures.

Audio or captions settings:

— **Hearing devices** - Includes the connections for hearing aids for people with hearing loss.
— **LED flash for alerts** - To enable people to choose what works best for them for notifications, particularly helpful for those with hearing loss.
— **Mono Audio** - Helpful for those with hearing loss.
— **Phone Noise Cancellation** - Reduces the ambient noise on phone calls when you are holding the receiver to your ear. Helpful for all, but also used by people with hearing loss.
— **Subtitles and Captioning** - For people with hearing loss, the background to the subtitles and the size of the text can be adjusted. Apple has also added a setting for hearing aid compatibility.

Other useful settings:

— **Shake to undo** - Users can shake the phone to undo for example, when typing.
— **Audio Descriptions** - For people with sight loss.

- **Guided Access** - This is helpful in education, or just where someone wants to limit what is accessible on the screen to a user.
- **Accessibility shortcut** - Set up to switch features on and off.

If you are working on iOS, make sure check out Apple's Developer area[20] and to follow Apple's accessibility guidelines[21]. These guidelines detail the API and provide an excellent source of hints and tips for maximizing the user experience with your apps.

Apple also provides some helpful guidance on testing the accessibility on your app with Voiceover[22]. Another useful link provides more information on how to make sure that the app reacts correctly to display settings[23].

New accessibility features in iOS

Apple are introducing new features to improve accessibility. These include:

- **Voice control** - Provides support to control the interface through voice as shown in the link below.

---

20 developer.apple.com/accessibility/ios

21 developer.apple.com/library/ios/documentation/UserExperience/Conceptual/
    iPhoneAccessibility/Introduction/Introduction.html

22 developer.apple.com/library/ios/technotes/TestingAccessibilityOfiOSApps/
    TestAccessibilityonYourDevicewithVoiceOver/
    TestAccessibilityonYourDevicewithVoiceOver.html

23 developer.apple.com/documentation/uikit/uiaccessibility

- **Disable autoplay** - This setting allows users to disable autoplay for videos extending the previous option (which applied only to certain apps) system-wide for all Apple apps and allows developers to extend this feature to third party apps.
- **Cross-fade screen transitions:** - When Reduce Motion is turned on, users will be able to change the way screen transitions are shown.
- **Differentiate without color:** - Replaces color indications with shapes or symbols that do not rely on color. It helps people with color blindness or monochrome vision.

iOS 13 added a new feature called Voice Control[24] enabling users to control their device through voice. This can convert commands such as swipe left to change the page. In addition, elements on the screen are numbered and the user can say "show numbers" and then, for example, the number related to the button they want to activate.

[24] support.apple.com/en-gb/guide/iphone/iph3c511340/ios

Windows App Accessibility

This section is included as Windows tablets contain some in-built accessibility features and it is useful to know about these when developing for this platform. A brief overview of the accessibility features is given below.

Speech output:

— **Narrator** - This is the screen reader for those users who are blind or have little vision.

Speech input:

— **Cortana** - This is the 'personal assistant', a main feature for all users, but will be helpful for those with disabilities too as it is speech activated.

Display settings:

— **Screen Magnifier** - This feature is for partially sighted people who wish to magnify the text on the screen and change the zoom level. It has its own gestures.
— **High Contrast Theme** - This theme changes text to black and white and provides a solid background behind words that would otherwise be on top of pictures. This is helpful for partially sighted users and some users with learning disabilities.
— **Text Scaling** - The size of text can be enlarged to aid those with learning difficulties or users who are partially sighted.

Touch or switch settings:

— **Activate keys on touch keyboard when I lift my finger off the keyboard** - This is to assist those with dexterity issues.

Audio or caption settings:

— **Closed Captions** - It is possible to change the font size, color, background and window transparency of captions. This is helpful for people with hearing loss that may also have some vision loss.

You can find out about accessibility for Windows apps with some Microsoft video resources[25] and platform-specific documentation.

Microsoft has published Guidelines for Designing Accessible Apps[26] and a dedicated paper about Accessibility for Windows 10 / UWP apps[27].

Mobile Web App Accessibility

As mentioned earlier in the chapter, much has been written about web accessibility, but less has been written on accessibility relating to apps. This is also true for mobile website accessibility or web app accessibility. It is an area which has growing interest and the World Wide Web Consortium (W3C) have created a Mobile Accessibility Task Force[28] concerned

25   developer.microsoft.com/en-us/windows/accessible-apps

26   msdn.microsoft.com/en-us/library/windows/apps/hh700407.aspx

27   docs.microsoft.com/en-gb/windows/uwp/usability/index

28   w3.org/WAI/GL/mobile-a11y-tf

with the required work in this area. In WCAG 2.1 there are additional principles aiming to address mobile access.

On the main W3C Mobile Accessibility page[29] you can find lots of helpful resources related to mobile accessibility.

It is suggested by the W3C that anything that uses HTML and is web-based should still follow the Web Content Accessibility Guidelines (WCAG) 2.0 while also referring to Mobile Web Best Practices (MWBP). So if you are a web content developer, then these guidelines are a good place to start. You will also find Relationship between Mobile Web Best Practices (MWBP) and Web Content Accessibility Guidelines (WCAG)[30] a helpful resource.

If your app is intended to mimic a native app look and feel, then you should follow the guidelines mentioned above in this chapter.

As support of HTML 5 is increasingly adopted on the various mobile platforms, consider reading Mobile Web Application Best Practices[31] as this is likely to form the foundation of any mobile web application accessibility standard that emerges in the future. One of the other key areas of guidance is Accessible Rich Internet Applications 1.0 (WAI-ARIA)[32], as it has been designed to ensure that more dynamic HTML functionality is accessible to screen readers.

An interesting area of work happening at the W3C is in the Independent User Interface (IndieUI) Working Group[33]. The group states "Independent User Interface (IndieUI) is a way for user actions to be communicated to web applications

29 w3.org/WAI/standards-guidelines/mobile
30 w3.org/TR/mwbp-wcag
31 w3.org/TR/mwabp
32 w3.org/TR/wai-aria
33 w3.org/TR/indie-ui-context

and will make it easier for web applications to work in a wide range of contexts — different devices, different assistive technologies (AT), different user needs". This work is going to be very important for accessibility and device independence. It is worth looking at the documentation that they currently have available.

Developing Accessible Games

Accessible games for disabled users were always very basic, if there were any available at all. Now some developers are beginning to think about engaging with a wider audience when developing games and finding ways to prevent unnecessarily excluding players.

Much of the guidance and standards already shared will be very valuable when creating games as this information is still relevant to gaming development. Do not forget to ensure that any accessibility features are clearly obvious to users in the game description so when they make the choice to purchase, they know that the app may be of interest to them.

There are a couple of sites that have some accessibility guidelines for game developers, e.g. *gameaccessibilityguidelines.com* and *includification.com*.

If you would like to look at some accessible games that have already been developed, then *game-accessibility.com* is a good resource.

Marc van't Veer & Julian Harty

Testing

We would like our apps to be well tested so we have confidence they will work and be positively received by their audience. Your needs and approach to testing range from lone developers testing an app themselves to mature, structured teams where there is a team of dedicated software testers who undertake much of the testing. This chapter aims to help you find effective ways to test your mobile apps that meet your constraints (what you are able to do given the resources and tools you have available) and context (including how your organization works, the needs of the app, and how the app will be used).

Other people can help you with the testing, particularly when you actively solicit their involvement and feedback. Similarly, you can help yourself be more effective though being competent with relevant software tools, testing techniques, and working effectively with data, including logs and analytics tools.

The rest of this chapter will go deeper into how professional app testing is structured and aims to help you use the available time for testing in a smart way.

Key Areas of Testing

When implementing testing into your projects, here are some key areas we recommend you include.

Organisational Awareness

A development team is usually embedded in an organization. This organization should be setup in a way that encourages sharing mobile testing knowledge across all the teams. If possible, the development team should be located together. The organisation's release heartbeat should fit the mobile app team's schedule (every day, week, sprint) and a mobile test policy should be in place that defines the testing framework and have a test strategy on different layers (multiple environments, scope, architecture, end-users).

Test Environment

Test environments are **where** we will perform the testing. They include many facets. ISTQB defines test environment as: "An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test."[1].

Your goal should be to have at least one environment that provides production-like data variations, settings, conditions and infrastructure which is comfortable to use for testers.

Often a testing environment includes special software tools and utilities, e.g. to read and filter log files, to control network behavior, to represent systems the app depends on including third-party authentication, to create test accounts etc. Often you will find solutions available already, but sometimes you might end up creating your own tools, for instance to mock the

---

[1] istqb.org/downloads/send/20-istqb-glossary/186-glossary-all-terms.html

behavior of an application server, to be able to inject errors, and so on.

Testers love to have multiple test environments from development all the way into production; this is called a "DTAP" phased approach: "Development, testing, acceptance and production environments"[2]. Each environment reflects a stage in the life cycle of any app and allows testing in each of these stages.

Test Device Policy

What a browser is for a website, is the hardware for a mobile app.

Platforms, networks, device settings, device models, movement, sensors and even firmware, are all specific and differ from each other. Any could cause problems for your applications[3]. This means we need a variety of devices to test on.

From a user's perspective a mobile app is the combination of software, hardware and environment. They want and expect an app to work, regardless of all these details. Furthermore, the boundaries of what needs to work extends even further, as many apps now interact with external devices and sensors (a subset of these may be useful for testing your app). As example, Disney have designed seamless experiences using smart wristbands[4] which integrate into a larger ecosystem. Devices can also be used to pay for travel[5] and shopping; cameras are used by banking software to identify and authorize cash withdrawals, and so on. Any problems, incompatibilities and

---

2   en.wikipedia.org/wiki/Development,_testing,_acceptance_and_production

3   An example of device specific problems with Android on Samsung is
    anasambri.com/android/special-place-for-samsung-in-android-hell.html

4   wired.com/2015/03/disney-magicband

5   vodafone.nl/shop/mobiel/abonnement/extra-opties/smartlife/wallet/reizen

so on, can adversely affect the UX and may have significant impact, for instance, if users are denied access to transport, money, and more.

A basic strategy for testing mobile apps is to assume that every combination is unique and different from another and may behave slightly differently, however, it would be impractical to test each combination. A more fruitful approach is to invest time in learning what the impact of the differences are, and testing a subset of the combinations that maximize the insights and confidence we have in the behavior of the app across as many of the combinations as practical. Key skills include:

— Device analysis: Which devices are the ones that are used among your target audience? What are the main differences? When and how are these differences relevant to the app? Which differences can be ignored?
— Extrapolation: What does a test on one device say to the thousands of devices out there in the wild?
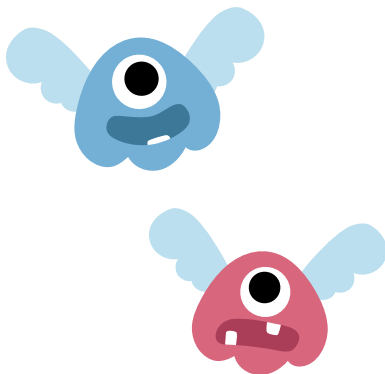
The set of test devices to use needs to be reviewed on an ongoing basis, as the app and the ecosystem evolve. Also, you may identify new devices, that your app currently does not support, during your reviews. The following figure illustrates these concepts.
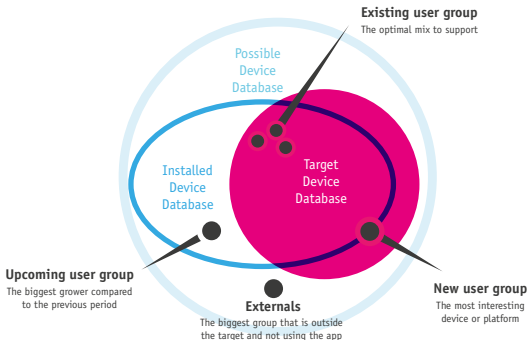
Complement your testing by using usage data gathered from production. Google provides developers with Google Play Console and Android Vitals which provide a wide variety of interesting and relevant data of an app's userbase and how well the app is liked and performs. Developers can also incorporate various analytics libraries in their app. These range from single purpose libraries for crash reporting or heatmapping to composite libraries that generally include mobile analytics.

Virtual devices

Virtual devices run as software, inside another computer, they are often free and immediately available to install and use. Some platforms, including Android, allow you to create custom devices, for instance, with a new screen resolution which you can use for testing your apps before suitable hardware is available. Virtual devices can provide rough-and-ready testing of your applications. Key differences include: performance, security, and how we interact with them compared to physical devices. These differences may affect the validity of some test results. Beside the Android platform virtual devices, you can use *GenyMotion.com*, who provide faster and more capable Android emulators, for instance to control the values of various sensors.

Physical devices

The performance characteristics of various phone models vary tremendously from virtual devices on your computer. So: buy, rent, beg or borrow phones to test on. A good start is to pick a mix of popular, new, and models that include specific characteristics or features such as: touch screen; physical keyboard; screen resolution; networking chipset, et cetera. Unless your target group only uses a certain type of phones, try your software on at least one low-end or old device as you want users with these devices to be happy too.

Here are several aspects to test explicitly on physical devices, as the devices have a significant impact on these aspects:

- **Navigating the UI:** For instance, can people use your application with one hand? Effects of different lighting conditions: the experience of the user interface can differ in real sunlight when you are out and about. It is a mobile device – many users will be on the move. Rotate the screen and make sure the app is equally attractive and functional.
- **Location:** if you use location information within your app: move – both quickly and slowly. Go to locations with patchy network and GPS coverage to see how your app behaves.
- **Multimedia:** support for audio, video playback and recording facilities can differ dramatically between devices and their respective emulators.
- **Internet connectivity:** establishing an internet connection can take an incredible amount of time. Connection delay and bandwidth depend on the network, its current strength and the number of simultaneous connections. Test the effects of intermittent connectivity and how the app responds.

Remote devices

If you do not have physical devices at hand or if you need to test your application on other networks, especially abroad and for other locales, then one of the 'remote device services' might help you. They can help extend the breadth and depth of your testing at little or no cost. Device farms are becoming commonplace and are clearly strategic where Google[6] and Amazon[7] in particular now provide them.

---

6   firebase.google.com/products/test-lab
7   aws.amazon.com/device-farm

You can also use commercial services of companies such as *SauceLabs.com*, *Testdroid.com*, *PerfectoMobile.com* or *Sigos.com* for similar testing across a range of devices and platforms. Some manufacturers brand and promote these services. However you often have to pay for them after a short trial period. Some of the commercial services provide API's to enable you to create automated tests. You can even create a private repository of remote devices, e.g. by hosting them in remote offices and locations.

Beta Testing

Another possibility, to increase the pool of devices, is to involve end users in the testing; for instance using beta testing. In this way the mobile app is on the devices and in the hands of end-users in their own environment. These devices are not controlled in any way. Extra tooling is needed to distribute the app and to capture the feedback from end-users. Check Ray Wenderlich's blog for more info on IOS beta testing[8] and Android's Developer portal[9] if you are targeting Android.

Mobile Testing Skills

Beside the basic skills like installing apps with Xcode and Android Studio it makes your life as a tester a lot easier if you know how to use the command line, analyse logfiles, create custom test data and to mirror apps[10].

Testers of mobile apps should understand the context of usage of the app, the types of users of the app;, know how to obtain device logs, know how to use analytics, be able to test API's directly, follow market developments of the OS, brands,

---

8   raywenderlich.com/5352-testflight-tutorial-ios-beta-testing

9   developer.android.com/distribute/best-practices/launch/test-tracks

10   See airsquirrels.com/reflector and vysor.io

and versions; and be able translate requirements from different perspectives into tests. Additionally, a tester should be familiar with the Agile/Scrum testing approach, risk analysis, test strategies and be able to use all sorts of tools during testing manually and automatically.

For professional QA training and certifications, you might want to evaluate the offering of the ISTQB (International Software Testing Qualifications Board)[11].

Beside these functional skills, testers should be able to deliver feedback effectively. Today's world of software development is characterized by agility: Apps are developed and released in small iterations (sprints) that reflect the constant improvement based on end-user feedback and other stakeholders' input. The tester is one of these stakeholders and needs to deliver his/her input in a way that is constructive and enables developers to implement improvements within a sprint cycle. In this setup, a tester is rather a coach on quality than the gatekeeper of quality.

To prevent feedback being received as negative criticism there are a number of guidelines to follow[12]. As a suggestion, try to make your team appreciate bug reports instead of perceiving them as criticism of their work.

A good way to constantly improve is pair testing. Pair testing is a technique in which two team members work together at one computer to test the software application. Pair testing can also be done together with a developer or designer[13].

---

11 See istqb.org for some general testing qualification paths and check their mobile specific offering at istqb.org/certification-path-root/mobile-application-testing

12 See teachthought.com/pedagogy/20-ways-to-provide-effective-feedback-for-learning

13 See stickyminds.com/article/many-advantages-pair-testing to learn more about pair testing

Apply Testing Techniques

To make sure your testing covers all relevant areas of your offering, a couple of mnemonics might help in memorising those:

— **I SLICED UP FUN**[14]**:** **I**nput (Test the application changing its orientation (horizontal/vertical) and trying out all the inputs including keyboard, gestures etc.), **S**tore (Use appstore guidelines as a source for testing ideas), **L**ocation (Test on the move and check for localization issues), **I**nteraction/Interruption (See how your app interacts with other programs, particularly built-in native apps), **C**ommunication (Observe your apps behavior when receiving calls, e-mails, etc.), **E**rgonomics (Search for problem areas in interaction, e.g. small fonts), **D**ata (Test handling of special characters, different languages, external media feeds, large files of different formats, notifications), **U**sability (Look for any user actions that are awkward, confusing, or slow), **P**latform (Test on different OS versions), **F**unction (Verify that all features are implemented and that they work the way they are supposed to), **U**ser Scenarios (Create testing scenarios for concrete types of users), **N**etwork (Test under different and changing network conditions)

14   kohl.ca/articles/ISLICEDUPFUN.pdf
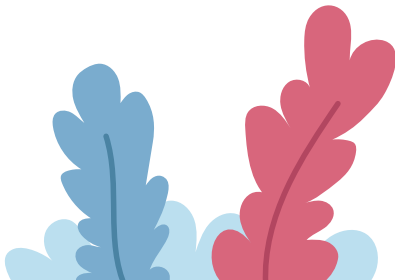
- **COP FLUNG GUN**[15] summarizes similar aspects under **C**ommunication, **O**rientation, **P**latform, **F**unction, **L**ocation, **U**ser Scenarios, **N**etwork, **G**estures, **G**uidelines, **U**pdates, **N**otifications.

Karen Johnson provides helpful material on using heuristics and mnemonics for testing software at *karennicolejohnson. com/wp-conteuploads/2012/11/KNJohnson-2012-heuristics-mnemonics.pdf*.

Tours

Tours are an experience-based test technique. It help us focus on **what** and **how** while we are testing a mobile app.

Tours help you focusing on your testing, Cem Kaner describes a tour as "a directed search through the program. Find all the capabilities. Find all the claims about the product. Find all the variables. Find all the intended benefits. Find all the ways to get from A to B. Find all the X. Or maybe not ALL, but find a bunch."[16]. With the combination of different tours in different perspectives (see the I SLICED UP FUN heuristics) coverage and test depth can be chosen.

Examples of Tours[17] include:

- **Configuration tour**: Attempt to find all the ways you can change settings in the product in a way that the application retains those settings.
- **Feature tour**: Move through the application and get familiar with all the controls and features you come across.

---

15 moolya.com/blogs/2014/05/34/COP-FLUNG-GUN-MODEL

16 kaner.com/?p=96; also see developsense.com/blog/2009/04/of-testing-tours-and-dashboards/

17 from michaeldkelly.com/blog/2005/9/20/touring-heuristic.html

- **Structure tour**: Find everything you can about what comprises the physical product (code, interfaces, hardware, files, etc.).
- **Variability tour**: Look for things you can change in the application - and then you try to change them.

Mobile Analytics

Mobile Analytics can help testers to identify differences in various aspects, e.g. performance and power consumption of a service's infrastructure when the app is being used by many users on a vast variety of devices. Some compelling examples of differences in behavior and on ways issues were addressed in a paper published by computer scientists from the University of Wisconsin[18].

Analytics is about finding trends in big numbers. This means to transforming raw data into meaningful patterns and act accordingly. To discover patterns in the collected data, there are (in general terms) four steps: 1. creation, 2. recording, 3. processing, 4. asking questions. More info about finding patterns can be found in another article of mine published as a PDF entitled "The magic of the mobile numbers: Testing with the power of analytics"[19].

18 "Capturing Mobile Experience in the Wild: A Tale of Two Apps", available as a download at static.googleusercontent.com/media/research.google.com/en//pubs/archive/41590.pdf

19 polteq.com/wp-content/uploads/2017/10/The-magic-of-the-mobile-numbers-Marc-van-t-Veer-2017.pdf

HP Enterprise also offers a book on the confluence between mobile analytics and testing for mobile apps[20].

See the chapter about Mobile Analytics to learn more about how analytics can help you assure your app's quality.

Users & Personas

Although the developers and testers might find an app easy to use, your end users might not have the same experience. The chapter "From Idea To Prototype" already explains how to design your app in a way that it delivers best possible user experience that fits your target groups' expectations and abilities. However, this process should - of course - never be considered as finished. To be able to continuously reflect on usability and get aware of possible issues, you should consider working with personas. Personas are fictional people that represent a certain type of end user. Putting yourself into their shoes helps understanding their needs and adjusting your software accordingly.

Learn more about the concept of personas at personas.dk and consider creating test cases especially tailored for these personas.

Involve End-Users in your Testing

Early and constant user feedback is an important mirror when developing any software - during any phase in development, including QA. It does not need a large budget or a huge number of users[21]. Significant value is gained with early involvement, diverse users, regular sessions and multiple smaller tests. Testers can guide and facilitate the end-user

---

20   themobileanalyticsplaybook.com

21   nngroup.com/articles/why-you-only-need-to-test-with-5-users

testing. For instance, by preparing the tests, processing log files, and analyzing results.

Whenever you include external people to test your app, they need ways to access and use the app. Web apps can be hosted online, perhaps protected using passwords, hard-to-guess URLs and other techniques. Installable apps need at least one way to be installed. For instance using a corporate app store or specialist deployment services like AppCenter (formerly known as HockeyApp)[22]. When the app is closer to being production-ready, users can test the more mature version of the mobile app in alpha & beta tests phases. See the platform chapters to get some hints on your options in that regard.

Crowd Testing

There are various services available that facilitate pre-production testing by external people, often so called micro-jobbers: Crowdtesting service providers include *Applause.com*, *PassBrains.com*, and *TestBirds.de*. *Lookback.io* provides a different more personal flavor. All of them usually offer you the option to define certain test cases or areas of your app as well as narrowing down the type of users you are interested in getting feedback from. Crowdtesting can also help to cover a wider range of real devices. But you should keep in mind that crowd-testers are often paid per bug. So do not trust their observations blindly. Always be able to validate them on local physical devices.

22 appcenter.ms

API Testing

An Application Programming Interface (API) is very often a key element of mobile offerings. They are abstracting a system's underlying implementation and only expose objects or actions a developer needs. As an example, a server API provides data to a mobile app in a format both have agreed upon. If the API does not deliver that data in the defined format or in a performative manner, the app will be perceived as buggy or slowly. Testing the API separately from the mobile app has many advances, like faster and easier business logic testing, ease of automation, earlier detection of defects and their root cause, creation of test data for app testing, easier reproduction of bugs and having a monitoring tool for production issues (regression).

To start (headless) testing the API directly can feel like opening up the box of Pandora with terms like JSON, REST, Mocks/Stubs, Proxy, HTTPS error states and debug logging. But also, a world of useful special tools opens up: Popular tools like Postman[23], Swagger[24] and Charles Proxy[25] will make your life as a tester a lot easier.

By helping providing stable APIs you will be helping delivering a more robust app experience.

To learn more about this topic, check *ApiEvangelist.com*[26], *ProgrammableWeb.com* and *Guru99.com/api-testing*.

---

23 getpostman.com

24 swagger.io

25 charlesproxy.com

26 especially apievangelist.com/2018/01/12/api-life-cycle-basics-testing

Manage your Testing Time

Testing as you have discovered can take many hours. Far more than your time allows. Particularly if you are close to a deadline such as a release date. There are various ways you can manage time spent in testing, in parallel testing can be made more interesting, rewarding, and more productive.

— **Reduce setup time**: Find ways to deploy apps quickly and efficiently. Implement mechanisms to provide the appropriate test data and configuration on both the mobile device and the relevant servers. Aim to have devices and systems 'ready to test'.
— **Reduce time needed for reporting & bug analysis**: Data, screenshots, and even video, can help make bugs easier and faster to investigate. Data can include logs, system configurations, network traffic, and runtime information. Commercial tools can record actions and screenshots to reduce the time and effort needed to report and reproduce problems.
— **Risk analysis**: You can use the risk analysis to decide how and when to allocate testing effort. Risks are hard to determine accurately by the tester or developer alone; a joint effort from all the stakeholders of the mobile app can help to improve the risk analysis. Sometimes the mobile app tester is the facilitator in getting the product risk analysis in place.

- **Scaling testing**: Increase the throughput of testing by scaling it. For instance using test automation, cloud-based test systems and more humans involved in the testing can help increase the volume, and potentially the quality, of the testing. Using static analysis tools to review code and other artifacts can also help the team to find and fix problems before the app is released.
- **Test automation**: Automate what you can automate but keep it simple. Start during design by assigning what should be automated on unit and API level. Prepare the mobile app with accessibility labels and select user scenario's that every user touches. Then start thinking about UI automation.
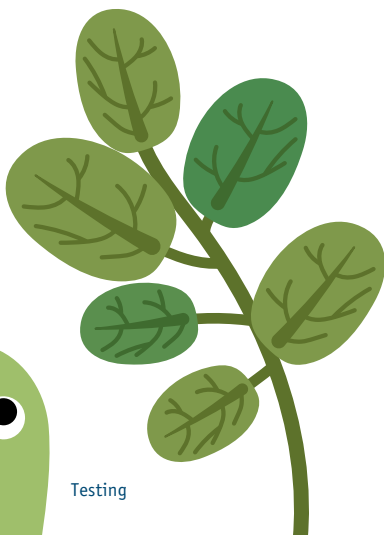
Testing

Defect Analysis

As stated above, the main purpose of testing is creating feedback, mostly based on the defects (aka bugs) found. A defect is a deviation from the expected behavior in the eyes of the user[27]. To be sure that a certain behaviour is unexpected, of course, a clear understanding of the expected behaviour is needed. This is one of the reasons where effective documentation can be particularly useful.

Any bug is a chance to learn and improve. To fix a bug completely you need to know its root cause(s) otherwise you may just fix a symptom or an isolated instance of a broader issue. Mobile defects often originate often from combinations of multiple factors such as OS versions, outdoor conditions, other apps, timing issues, end user interaction, unexpected data input, limited memory, server outages etc.

The process of finding the cause is called defect analysis.

Taxonomy

Once you start researching, you will see that there are complex studies available on how to detect the root cause of a bug. Many people have graduated on this topic. In Ajay Kumar Jha's work on defect analysis[28], he created a repository, a taxonomy of possible risks. If these risks becomes reality, then there is a defect in the mobile app. His idea behind the taxonomy defect analysis method is to "define feature categories and collect lists of possible bugs in each category".

A good place to build a taxonomy, is the app store reviews.

---

27 Find a short explanation about other defect definitions at getzephyr.com/insights/what-is-a-software-defect

28 "A Risk Catalog for Mobile Applications": testingeducation.org/articles/AjayJha_Thesis.pdf

Both Google's Playstore and the iOS App Store lets you export all reviews[29] which you can then use as a basis for your defect analysis.

A defect taxonomy may help you to create better tests, see Cem Kaner's[30] and Michael Stahl's online articles[31] to learn more.

S-FMEA

A second approach for analyzing defects is S-FMEA (Software Failure Mode and Effect Analysis[32]). The S-FMEA method is looking for structural problems in a system design. First you define possible failure modes, then you examine its consequences on different system levels as part of the effects analysis. S-FMEA can support the decision-making process during the architectural planning of a system. This means that there are alternatives choices and each choice can lead to different failure modes of the application and have different effects. Understanding this relationship and making the right decisions can lead to a more robust design and mobile app.

---

29 check support.google.com/googleplay/android-developer/
    answer/138230#export_ratings_and_reviews and help.apple.com/app-
    store-connect/#/devd15088dd0 elp.apple.com/app-store-connect/#/
    devd15088dd0 to learn how

30 semanticscholar.org/paper/Bug-Taxonomies%3A-Use-Them-to-Generate-
    Better-Tests-1-Vijayaraghavan-Kaner

31 stickyminds.com/article/using-bug-taxonomy-design-better-software-tests

32 See "An Introduction to Software Failure Modes Effects Analysis (SFMEA)":
    slideshare.net/AnnMarieNeufelder/an-introduction-to-software-failure-
   modes-effects-analysis-sfmea

Learn More

Testing mobile apps is becoming mainstream with various good sources of information. Useful online sources include:

— *katrinatester.blogspot.de/2015/08/mobile-testing-pathway. html* - A comprehensive and well-presented set of possible steps for testing mobile software.
— *github.com/julianharty/testing-heuristics* - An online open source project to learn more about testing heuristics for mobile apps.
— *enjoytesting.files.wordpress.com/2013/10/mobile_testing_ready_reckoner.pdf* - Contains short, clear testing ideas with examples, mainly for Android devices
— *developers.google.com/google-test-automation-conference* - The annual Google Test Automation Conference (GTAC) often includes several presentations on testing mobile apps. These are recorded and available free of charge. Worth watching.
— *genymotion.com/blog/android-testing-showdown* - A useful guide on selecting the best devices to test on.
— *appqualityalliance.org/resources* - The official App Quality Alliance AQuA website including their useful app testing guidelines.

A good place to start learning testing mobile apps is reading books like:

— Tap Into Mobile Application Testing by Jonathan Kohl: *Leanpub.com/testmobileapps*.
— Hands-on Mobile App Testing by Daniel Knott. A sample chapter is available at *HandsOnMobileAppTesting.com*.
— Testing Computer Software by Cem Kaner
— Steve Krug has written several immensely popular books that cover low-cost DIY usability testing. Various chapters are available online at *Sensible.com*. His work is well worth reading and much applies to testing mobile apps.

Also make sure to read the Security chapter in this guide: It contains valuable guidelines how to test your app and its components regarding vulnerabilities. And of course, do not miss the platform-specific articles (especially the mobile web chapter) to gain deeper insights into mobile testing.

BY Linda Harnisch
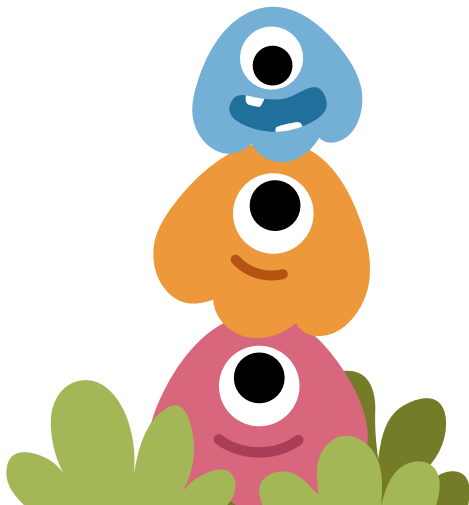
Monetization

For developers, mobile has democratized access to billions of users. According to Sensortower[1], in 2018 more than 71 billion USD were transacted in major app stores of Apple and Google with in-app purchases, subscriptions, and premium apps; up from 58 billion USD in 2017. Just take a look at the most profitable (non-gaming) mobile apps[2] across the App Store and Google Play Store such as Tinder, Netflix or YouTube to notice that users do not mind paying for apps. There is an incredible opportunity for app developers to reach consumers; and of course there is a huge competition with over 5 million apps[3] in the leading app stores (Apple's App Store and Google Play). Over 94 percent of apps on the main app stores are free[4] and compete for users' attention. Thus, developers need to make sure that their app and their advertising budget help their app stand out from the crowd.

1   sensortower.com/blog/app-revenue-and-downloads-2018

2   sensortower.com/blog/top-grossing-apps-worldwide-may-2019

3   statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores

4   statista.com/statistics/266211/distribution-of-free-and-paid-android-apps

Monetization Models

Assuming you are not developing your app as a hobby, you probably want to turn it into a source of revenue. This is where app monetization models come into play. This should be incorporated into your business plan well before the launch of your app.

This chapter will explore dominant app economy monetization models that are somehow dependent on the main app stores:

— In-app purchases: Add payment options into your app.
— Subscription Models: Gain recurring revenue by gating access to content and features.
— Pay per download: Sell your app per download.

Additionally, we will also discuss other monetization options that are less dependent on platforms:

— "Freemium" with in-app purchases: The app is free to download but includes premium features that users have to pay to access.
— Mobile advertising: Earn money from advertising.
— Sponsorships: Receive money for each user signing up to your sponsor.
— Data Monetization: Mine your mobile app user data.
— Indirect sales: Physical goods among others.

When you come to planning your own development, determining the monetization business model should be one of the key elements of your early design as it might affect the functional and technical behavior of the app.

Considerations For Your Monetization Strategy

Ensuring your mobile app is economically viable requires defining a monetization strategy from day one. Choose an app monetization strategy that will allow you to profit off your development efforts and costs.

App Vertical

Some verticals lend themselves to certain forms of monetization. What problem does your app solve and which service does it provide? If you are offering a content centered service such as a music or video streaming service (think Spotify or Netflix), a subscription model will be profitable. Freemium models, on the other hand, are lucrative for apps that want a mix of revenue such as ads or in-app purchases (think gaming apps).

Platform

One of the factors that impact your revenue is the platform you are developing for. Whether you are developing for iOS or Android (or both) impacts your monetization capabilities due to market penetration and platform popularity. Looking at the two major app stores, Apple is generally the platform that is seeing maximum revenue growth. Basically twice as much as what Google Play Store is making in terms of in-app purchases, subscriptions and premium apps. According to data by Sensortower, the stores saw a combined 19.5 billion USD in consumer spend in the first quarter of 2019; users of Apple's App Store spent 12.4 billion USD and Android users 7.1 billion USD in the Play Store[5].

5   sensortower.com/blog/app-revenue-and-downloads-q1-2019

Competitors

Research what competitors in the same app vertical are doing. How do they monetize; and is their chosen approach working for them? Maybe you will find a gap or a niche that you can fill.

Main Revenue Stream

As mentioned in the introduction, you need to evaluate whether your app is your main product and revenue stream. Or, whether it is just another channel or vehicle of your actual product. Is your mobile app an extension to your existing PC web shop or physical store? Offer the app for free and earn revenue from your products and services in the real world (and save the 30 percent revenue share from Apple & Google).

User Acquisition

Of course, app monetization is not possible if you do not have a significant volume of app installs. You need to have a large amount of MAUs (monthly active users) to earn sustainable revenue. This definitely holds true for the in-app advertising model. This is where user acquisition (UA) strategies step in. With this edition, we added a dedicated chapter about user acquisition to our guide. Make sure to read it carefully.

User Experience

Your monetization strategy should complement your app's user experience and fit organically within the app. If you put ads into your app, make sure that this is seamlessly done. You will want to choose a marketing partner/network that supports interactive ads as well as analytics and targeting.

Value Proposition & Pricing

In the ASO chapter of this guide you will learn how to create an app store listing that communicates the value of your app. It will help you convert app searchers into paying users if you clearly communicate your value proposition.

If you are choosing the pay per download model you should offer an attractive price tag. According to Statista, the average price in 2019 for an app on the App Store is 1.01 USD[6].

Monetization - Platform Dependant

In-App-Purchases (IAP)

In-app purchases (IAP) are a way to charge for specific actions or items within your application, whether physical or virtual in nature. You can offer the basic features of your mobile app for free, but charge for premium content and premium features. For example, virtual currency or additional levels, videos, premium information or removing ads. This monetization approach is used by some of the top grossing apps[7] in gaming, e-commerce or dating apps such as Tinder, Evernote, Candy Crush or Clash Royale.

According to AppsFlyer[8], the average mobile app user spends 0.50 USD per month on IAP per app. The average monthly in-app spend for gaming apps is 0.32 USD; for shopping apps it is 2.68 USD. But with only 5 percent of users actually spending money on IAP, that does not sound like a promising way to earn money with your app. The challenge lies

---

6   statista.com/statistics/267346/average-apple-app-store-price-app

7   statista.com/statistics/271674/top-apps-in-google-play-by-revenue

8   "New Report on Global In-App Spending Habits Finds That Asian Consumers Spend 40% More In Apps Than the Rest of the World": bit.ly/290JNvi

in acquiring an active enough user base where that amount of money turns into a sustainable revenue for your app.

Most app stores offer an in-app purchase option, or you can implement your own payment mechanism. It should also be obvious that you will need to design and develop your application to incorporate the in-app payment method. If your application is implemented across various platforms, you may need to implement a different mechanism for each platform (in addition to each app store, potentially).

As with pay per download, we would recommend that you start with the in-app purchasing mechanism offered by an app store, particularly as some of these can leverage operator billing services (such as Google Play) or utilize pre-existing credit card information (such as Apple or Amazon), or with in-app payment offered directly by operators. From a user's perspective, this is the easiest and most convenient way to pay, so developers can expect the highest user acceptance and conversion rates.

Subscription Models

Monetizing apps through subscription is gaining momentum. In-app subscriptions are a subset of Freemium/IAP, essentially letting users download an app for free initially. Users can then access a limited amount of content before being prompted to pay for premium content or features. Subscriptions are lucrative for apps that are content-driven such as newspaper-, dating-, music streaming and utility apps. Spotify's[9] freemium success story is often quoted as the best example[10] of an app for turning freemium into a billion-dollar business with subscriptions.

Subscriptions take IAP to the next level. Instead of paying only once for an upgrade, users make regular payments bringing in steady revenue. By continuing their subscription, users are encouraged to interact with the app. This boost in user retention rates is quite profitable to cover long-term development costs. On the other hand, it means you want to keep user churn low. You can read more on preventing users to churn in the "User Retention" chapter.

The pricing structure of subscription models has become more complex and varied. In fact, they mirror the pricing model of a SaaS (Software as a Service)[11] by now. App developers charge users either a monthly or yearly fee for the app and offer multiple subscription options, as well as an option that combines all the services in a premium package.

In-app purchases and subscriptions for your mobile app are both managed by Apple (iOS) and Google (Android). In order to set up subscriptions in the app you should implement

---

9   spotify.com

10   "Spotify's leadership is doubling down on an unprofitable strategy", bit.ly/2YicZDy

11   "5 App Monetization Trends To Watch In 2018": bit.ly/2LGMGB9

Apple's In-App Purchase API[12] or the Google Play Billing API for Android[13]. Apple is especially pushing for subscriptions by offering incentives for developers. For the first year of any user's subscription, Apple will get 30 percent of the App Store purchase. Google is taking a 30 percent cut of developer's revenue as well. In 2016, Apple changed its policy to take only 15 percent of the marketplace fee once customers subscribe for more than a year. Yet, some brands such as Netflix[14], Epic Games and lately Tinder[15] have decided to bypass the stores, instead directing their users to the web or other platforms in order to avoid fees - with mixed results[16].

In September 2019 both major mobile platform providers launched their very own subscription services: Apple Arcade offers exclusive access to games while Google's Play Pass also includes apps.

Pay Per Download (PPD)

Pay per download used to be the dominant monetization strategy for a long time. The approach is simple: Sell your mobile application upfront charging per download. Payment can be handled by an app store or mobile operator.

Using PPD can typically be implemented with no special design or coding requirements for your app. For starters, we would recommend using the app store billing options as

---

12 developer.apple.com/app-store/subscriptions

13 developer.android.com/google/play/billing/billing_overview

14 "Netflix stops paying the 'Apple tax' on its $853M in annual iOS revenue", tcrn.ch/2F2eVFW

15 "Tinder is now bypassing the Play Store on Android to avoid Google's 30 percent cut", bit.ly/2OkFSuK

16 mobiledevmemo.com/spotify-vs-apple

it involves minimal setup costs and minor administrative overhead.

As the mobile economy and monetization strategies have matured over the years, today only 3 percent of non-gaming app revenues are generated through paid app downloads[17]. Aggregated data by Statista shows that global mobile app revenues through PPD have experienced a slowdown for a number of years by now[18]. Pay per download generated about 29 million US Dollars in 2017; compared to IAP accounting for about 37 million USD[19].

The benefit of the paid-app approach is that developers earn revenue upfront. User retention or lifetime value do not carry such a big weight. Yet, as pointed out above, the majority of apps these days are available for free. Users find it a standard practice to not pay for an app product that they have not even tried out yet. The tendency for PPD can hardly be called strong or profitable for app publishers nowadays.

Monetization - Platform Independent

Freemium
The majority of apps monetize via freemium with the ability to unlock premium features with in-app-purchases (IAP) or through other channels. Think of Skype, Spotify or LinkedIn. Apple introduced in-app purchases in 2009 and since then this model has become a tried-and-proven mobile app monetization strategy.

---

17 statista.com/statistics/273120/share-of-worldwide-mobile-app-revenues-by-channel

18 statista.com/statistics/273122/global-paid-for-mobile-app-revenues-forecast

19 braze.com/blog/in-app-purchase-stats

Why has freemium become so dominant? Unlike pay per download (PPD), the freemium model appeals to users' desire to test an app before buying it. It allows you to hook a good deal of first-time users before charging them while trust is built. You have to engage users enough before turning them into buyers who are willing to spend for a more increased experience. Therefore, gaining revenue from freemium depends very much on user engagement.

In-App Advertising

Generating revenue by displaying advertisements in your mobile app has become a viable method for driving conversions. Once you have decided to advertise with ads you have to choose an ad network[20] to monetize your app and add its SDK to your app. Each network offers slightly different approaches specializing in individual ad formats, pricing models, geographic regions and advertiser types.

Most often the ad networks' revenue for your app does not scale at the same rate as your users base does. There will be issues with monetizing all the different geographies or different ad types. You will soon realize that though you are generating revenue from your user base, you are introducing problems around CPM (we will explain this payment model) and keeping the competition with different ad sources you are working with. Thus, instead of working with one ad network SDK, you have to start working with several SDKs as your users continue to scale. That is a lot of SDKs to manage and may even slow down your app and affect performance. The complexity within your app will be huge.

Ad mediation platforms such as *AdApptr.com* or *MoPub.com* can can help you optimize ad revenue and solve some of the

---

20 See Ryte Wiki definition of Ad Network: en.ryte.com/wiki/Ad_Network

problems of scaling your ad monetization. As you are working with a mediation platform you have one single mediation SDK that gives multiple different ad networks access to your inventory and then gets them all to compete. Now you only have one simple integration making sure that the highest bidder wins by comparing what CPM all the various ad networks are offering. Working with mediation is the way to go, to get the most out of your inventory, optimize your CPM and increase your ad revenue.

The app advertising industry keeps evolving and so do ad formats that app developers can choose from:

— **Banner ads** (static or animated)**:** Typically text or images that can be placed at the top/bottom of the screen. These are probably the most familiar ads for users but can be obnoxious.
— **Interstitial ads:** A full screen ad, which typically has a "skip screen" button at the bottom. Commonly used during transitions in the app. They come in a variety of formats, as text, image or app install ads.
— **Expandable ads:** Rich media ad combining banner and interstitial ads. Users can choose if they want to view the ad.
— **Native ads:** Ads that match the look and feel of your app and its content. Hence, the least intrusive ad format that offers users relevant content.
— **Video ads:** Ad format that can be highly engaging and shows key app features. Costly to produce and place, hence you should get the video ad right.
— **Rewarded ads:** An ad that rewards users for watching the ad.

- **Playable ads:** Offers users a taste of gameplay inside the app. Usually, there is a CTA[21] at the end prompting users to download.
- **Offer Wall:** Offerwalls are placed inside an app listing multiple offers that provide users with incentives in exchange for completion of specific actions.

If you are opting for ads, consider the placement in the app. Users are accustomed to ads in their mobile apps but you should nevertheless put emphasis on user experience. Avoid intrusive or irrelevant ads that can damage user retention.

If in-app advertising is your chosen strategy, decide what the objective of your ad campaign is. Usually advertisers decide between branding and performance marketing campaigns. The big difference being that: performance marketing campaigns aim for an actual action or results, e.g. installing an app; versus, growing public knowledge of your app through brand marketing. Both approaches call for different pricing models. Today, the most used mobile advertising pricing models are:

CPM (cost per thousand impressions)

The price an advertiser pays to an app publisher to serve 1.000 impressions of an ad. CPM is beneficial for app publishers and developers since they generate revenue every time an ad is shown. On the flipside, this presents a drawback for advertisers: they are charged regardless of whether anyone clicks their ad. As an advertiser, you want to ask yourself whether you are willing to pay just for impressions, not specific user action goals. If you are running brand awareness campaigns, CPM is the way to go. If you think about it, this is probably the fairest way of doing advertising. Many aspects that will

[21] Call to action

decide whether a user clicks or even downloads the advertised app are well outside of the control of the developer/owner of the ad inventory.

CPC (cost per click)

The price an advertiser pays each time a user actually clicks on an ad. CPC is a pricing model utilized in user acquisition (UA) campaigns. The advantage is that advertisers pay nothing for ads that do not generate clicks. However, as an advertiser you get charged for errant clicks that do not result in a lead or customer action. The drawback for app publishers presents itself in potentially serving ad impressions for free.

CPA (cost per action)

With this pricing model, the advertiser pays for clicks on their ads that lead to an action by a user (purchase, sign up, app download, subscription). This is beneficial for advertiser's UA campaigns since they only pay for specific outcomes. The drawback and risk for app publisher and developer: Serving ad impressions and generating clicks without actually generating a conversion (meaning they do not get any money).

CPI (cost per install)

The price an advertiser pays whenever the consumer installs the advertised application. Hence, this pricing model is especially used in UA campaigns. Advertisers find this model favorable since they only pay for users that install the app after seeing an ad promoting it. CPI emphasizes app down-loads and installs so the advertisers get what they pay for.

Yet, there are other creative ways of monetizing your app through mobile advertising. Don't constrain yourself and explore other popular approaches to boost your revenue.

Ad-Free Sponsorships

The sponsorship approach involves collaborating with brands who grant users rewards upon completing in-app actions. The sponsorship is integrated into the app which means no ads are displayed. When rewards are redeemed, app developers will get part of the revenue. Take *Apponsor.com* for instance. The user gets your app for free and is prompted to sign-up for a newsletter of your sponsor. In return, the sponsor will pay the developer an amount for each newsletter registration. It kind of works like an affiliate app program.

Data Monetization

As an app developer, you have access to an incredible amount of user data. This is a goldmine for digital and offline businesses alike. It reveals insights about consumers that these businesses can use to analyze their audience or product improvement.Yet, you need a great amount of daily active users (DAU) to employ data monetization at all. So-called data collectors will only get non-personal information such as device type or device version. We especially recommend these articles to learn more about data monetization, how to gather it correctly while also being trustworthy to your app users:

— "Reasons Why Data Monetization Should Be In Your 2019 App Monetization Strategy" (The Tool): *bit.ly/2ObKuDt*
— "App Monetization: Data vs. Ads" (Hackernoon): *bit.ly/2JULRka*

Indirect Sales

Another option is to use your application to drive sales elsewhere. Use your app as a marketing tool to sell goods in the real world. Typical examples are car apps, magazine apps and

large brands such as McDonald's and Starbucks. Also, coupon applications like Groupon often use this business model.

Final Thoughts: What Can You Earn?

Certain types of apps have made their developers millionaires (or even billionaires); many others are not generating enough revenue to break even with development costs. So one of the most common developer questions is how much money they can make with a mobile app and which monetization model works best. Many apps mix a variety of monetization strategies to create one that suits them best and goes in line with their product. Success usually comes with a degree of experimentation and a lot of perseverance.

Without a doubt, there is big money in apps. In 2020, statistics forecast apps to make a total global revenue figure of over 188 billion USD[22]; up from 88 billion USD in 2016. Gaming apps dominate the list of top-grossing apps. Developers such as King Digital Entertainment (Candy Crush Saga) generate over 93 million USD[23] in revenue through IAP from their worldwide Android game downloads alone. There are certainly other app verticals with impressive earnings. App publisher IAC reached over 260 billion USD[24] across Apple's App Store and Google Play Store in 2019 with dating app Tinder's subscription model[25].

So, how much can developers earn? The answer to this

22 statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast

23 statista.com/statistics/692559/leading-mobile-games-publishers-google-play-world-revenue

24 "Tinder becomes the top-grossing, non-game app in Q1 2019, ending Netflix's reign", tcrn.ch/2KqWcZb

25 sensortower.com/ios/us/tinder-inc/app/tinder/547702041

question depends on a lot of different factors such as size of development studio, location, coding experience and mobile platform choice (iOS or Android). According to the developer salary guide by Business of Apps, an average annual salary of a mobile app developer in the US is 107,000 USD, 47,000 USD in Germany and significantly lower in India with 4,100 USD[26]. Keep in mind that the local economy plays a significant part in the earnings though.

Last but not least, app developers want to consider monetization beyond mobile devices for future endeavours. There is an emerging market for additional smart devices and platforms. Think smart TVs, smartwatches and video game consoles to watch in future growth and monetization opportunities.

26   bit.ly/2Sy2R3X

Learn More

— "The Ultimate Guide To Mobile In-App Purchases Optimization": *instabug.com/blog/mobile-in-app-purchases*

— Business of Apps published a guide on top mobile ad networks 2018: *businessofapps.com/guide/top-mobile-ad-networks*

— Many articles are out there tackling the question of how much you can earn with apps. Such as this Medium article, "How Much Money Can You Earn With an App?": *medium.com/@fueled/how-much-money-can-you-earn-with-an-app-981667aed26e*

— A guide for publishers who want to monetize an Android app with AdMob and aren't/are using Firebase: *developers.google.com/admob/android/quick-start*

— If you are interested in monetization case studies, mediation platform MoPub offers some educational resources at *mopub.com/edu-center*

BY Laura Spikermann

App Store Optimization (ASO)

Once your app development is completed and you have created something incredible, it is time to present it to the public. After spending all this time developing your app, you want it to be found, recognized and used by as many people as possible. The market success of an app is often measured by the number of downloads or installs, which makes this KPI (key perfomance indicator) an important growth metric. To generate a significant number of downloads, your app needs to be visible to potential users. But just publishing it to the app stores will not do the trick here. The number of apps in the stores is constantly and rapidly increasing, which means your app is competing with more and more apps to be found by potential users. According to statista, Android users can choose between 2.4 million apps in the Google Play Store nowadays[1]. iOS users have slightly fewer apps to pick from, but with 1.9 million, it is still an overwhelming number. Thus, mobile app marketers and developers need to make sure that their apps stand out from the crowd. This is where App Store Optimization (ASO) comes into play.

While reading this chapter, you will find some tips and tricks on how to position your app in the best way possible, and therefore ensure organic growth.

---

1 statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

What is ASO?

App Store Optimization describes the process of maximizing organic app downloads or installs coming from high-quality users, by increasing visibility and conversion rate in the app stores. Searching through the app stores is the most popular way for users to discover new apps. According to Apple, 65% of iOS apps are discovered directly through search on the Apple App Store[2]. On the Google Play Store, the share is almost as high with 53% of downloads, according to AppRadar[3]. This underlines the importance of high visibility.

To make it simple, ASO helps you to attract more users by increasing the visibility of your app in the app stores and subsequently increasing the likelihood of converting users from seeing your app into downloading it.

Goals: Visibility and User Conversion

Users visiting the app stores usually decide on an app very quickly. When entering either the Google Play or Apple App Store, users have the option to find an app by searching with specific keywords or search terms, or browsing through the feature cards, categories, and lists of recommended or similar apps. The apps on top of the search results page are the ones most likely to be downloaded. Great ASO strategies will help you secure such a position for your app, and therefore become more visible to a large amount of potential new users.

But there is more to ASO than increasing the visibility. Many users who find an app still do not download it. It could be because the app ranks for irrelevant keywords that do not

---

[2]   searchads.apple.com/

[3]   appradar.com/academy/what-is-app-store-optimization

match the user's download intents. Or, it could be because the store listing of your app is not attractive or persuasive enough. This leads to the second mission of ASO, conversion rate optimization. This is especially important when investing in paid media activities to promote your app.

Thus, every user who wants to use your app needs to download it from one of the app stores first, so they need to visit your app store page. To ensure a high conversion rate, your app store page should be visually pleasing as well as textually compelling and provide the user with relevant information about the values and features of your app.

Platform Differences

When talking about ASO, it is important to have an overview of the different elements which can be optimized, and understand the differences between the Google Play Store and the Apple App Store in which your apps are published.

Google Play Store

Let us start off with Android and the metadata presented in the Google Play Store. There are seven main on-page elements to consider when optimizing the appearance of your app in the Google Play Store:
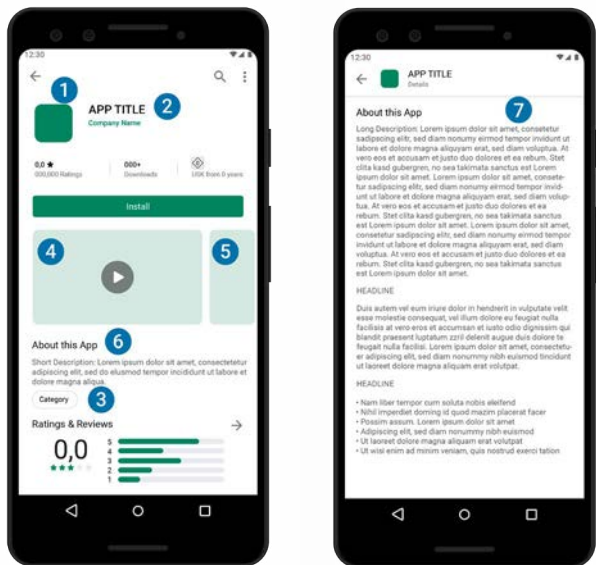
1. Icon
2. App Title
3. Category
4. Promo Video and Feature Graphic
5. Screenshots
6. Short Description
7. Long Description

App Store Optimization (ASO)

All of these elements except the promo video are mandatory and have to be provided when releasing your app to the Google Play Store via the Google Play Console. Unlike the Apple App Store, you are free to change the metadata and visual assets whenever you desire without having to publish a new version of your app. Something else which is unique for the Google Play Store are the Store Listing Experiments[4]. Those give you the option to A/B - test different variations of your app store page elements on the Google Play Store to identify the best performing combination based on downloads. When doing ASO for Android apps you should definitely take advantage of this testing possibility to analyze the impact of your changes.

[4]   developer.android.com/distribute/best-practices/grow/store-listing-experiments
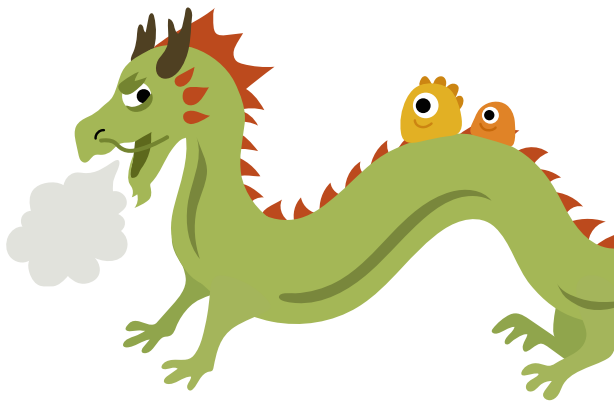
Apple App Store

For iOS apps, the structure of the app store page is rather similar, with a few major differences to note. In the Apple App Store, you have eight elements to consider for your ASO activities.
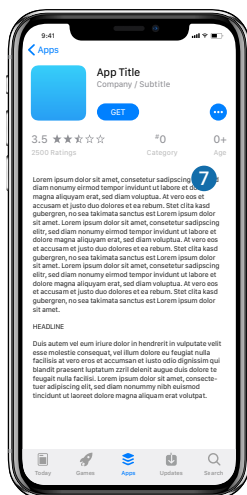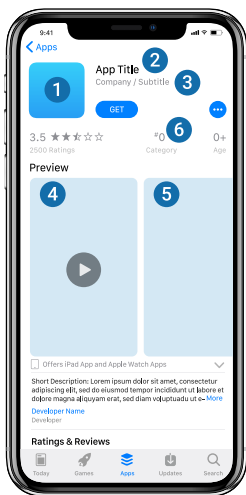
1. **Icon**
2. **App Title**
3. **Subtitle**
4. **Preview Video**
5. **Screenshots**
6. **Category Ranking**
7. **Long Description**
8. **Keywords (not visible)**

Two of these elements, namely the app preview video and the subtitle are optional, while the rest are mandatory. Another specialty in the Apple App Store is the keywords field, which is invisible for store visitors and competitors and can only be viewed and adjusted in the App Store Connect Console. Unlike the assets in the Google Play Store, changes in the Apple App Store can only be applied when releasing a new version of the app. Therefore, aligning your ASO activities with the development activities is crucial. The single elements and their impact on the visibility and conversion rate will be discussed in greater detail in the sections below.

Optimizing Textual Elements

One important aspect of ASO involves optimizing the textual elements (app title, subtitle, short description, long description, keywords field) of your app store page in terms of keyword usage. Using the right keywords in the right positions will boost visibility by helping your app to appear on top of the search results page for all relevant search terms. In order to boost conversions and therefore downloads, it is important to review if the selected keywords drive the right people towards your app and make adjustments if needed.

Keyword Ranking Algorithm

Each of the app stores has its dedicated algorithm which is responsible for ranking an app for specific keywords. The ultimate goal of every search ranking algorithm is to help users find the best products & services for their needs. This is done by showing a sorted list of results that best match the used search terms. The goal of ASO is to trigger the algorithms to consider your app as relevant for specific search terms.

The algorithm of the Google Play Store is indexing every textual part of an app store page. Therefore, the app title, short description and the long description should be filled with relevant keywords. The algorithm of the Apple App Store is more selective and only a few elements such as app title, subtitle and the off-page keywords field are indexed. Therefore, it is especially important for the Apple App Store to position your keywords wisely by placing the most relevant ones in the app title.

Keyword Research and Analysis

Keyword research is one of the most crucial ASO activities. Keywords are a decisive aspect for your app to be found on the Apple App Store and Google Play Store. One of the first steps in App Store Optimization is to perform a detailed keyword analysis.

There are different tools enabling you to see which keywords are valuable and which should possibly be replaced, for example:

— *AppTweak.com*
— *TheTool.io*
— *MobileAction.co*
— *AppRadar.com*
— *AppAnnie.com*
— *SensorTower.com*

Important metrics to focus on when analyzing your keywords are the search volume and the competition each keyword sees in the respective app stores. The goal is to identify keywords that will drive relevant traffic to your app store page and are not already under high competition. Keep in mind that a keyword analysis and ASO in general are typically

things you should repeat every once in a while in order to react to changes either in the store algorithms or in your app. The keywords you are using in the metadata of your app store page should be monitored and iterated regularly to achieve the best results.

App Title

Keywords in your app title have a lot of weight in the algorithm of both app stores. Therefore, try adding your most valuable and most relevant keywords in your app title to increase your app's discoverability. Choose an app title that contains relevant keywords, but still fits the purpose of your app and the guidelines of your brand. It should be a simple, memorable title that hints at what your app does. The best practice to add additional keywords to your app title is to put your app name first and then use a ":" or "-" to fill the remaining characters with relevant keywords. Apple allows a maximum of 30 characters for the app title while the limit on Google's Play Store is 50 characters.

Do not change the app title too often, as it may have a negative impact on your ranking and confuse your users. Please note that the app title in the app store is independent of the name of your app on the home screen of a device once it is installed. This means adding additional keywords to your app title in the app store will not change the name of your app once it is installed on the device in any way.

App Subtitle (iOS only)

The subtitle appears under the app title in the Apple App Store and allows you to explain the value of your app in greater detail. A subtitle does not exist in the Google Play Store. Keywords from the subtitle are indexed by Apple's algorithm and are therefore an important ranking factor. Avoid generic descriptions such as "World's Best App". Instead, highlight the unique value propositions of your app. As mentioned above, the subtitle is one of the few elements in the metadata of your app store page which is not mandatory. If you do not choose a dedicated subtitle, the publisher's name will be shown underneath the app title. This is not recommended since it does not let you make use of the allowed 30 characters that could be a substantial resource for visibility improvement.

Short Description (Android only)

The short description is only available for your app store page in the Google Play Store, and appears underneath the screenshots and promo video. Within the short description, you can motivate the user to download your app by providing a short and eye-catching one-line description, which explains the best features and functions of your app. Other approaches which could also be used for the short description are a call-to-action (CTA) (e.g. "Join the largest film-making community now!", or a user-centric trigger (e.g. "Can you solve the mystery?").

Depending on your app, one approach may work better for your audience than the others. This is something you could test with the Store Listing Experiments[5] which were mentioned above.

Short descriptions are limited to 80 characters, and they can contain emojis and rich text (HTML).

Long Description

Within the long description you can highlight features and functions of your app and guide the user on how to access the full potential of your app. Let potential users know what makes your app unique and why they will love it. The long description is only indexed by the algorithm of the Google Play Store, and not in the Apple App Store. When optimizing your long description for the Google Play Store, use your relevant keywords as often as possible while maintaining a meaningful and relevant content. For the long description, a good structure is vital. Subheadlines are a good way to enable interested users to quickly find relevant information. To keep it simple, you can reuse the content from your optimized long description from the Google Play Store for the Apple App Store. Keep in mind that changes on the Apple App Store can only be applied when publishing a new version of your app, while changes on the Google Play Store are possible at any time.

Apple as well as Google have set a limit of 4,000 characters for long descriptions. In Google's Play Store you can also use HTML formatting and emojis.

[5] developer.android.com/distribute/best-practices/grow/store-listing-experiments

Promotional text (iOS only)

The promotional text is an optional element on your app store page in the Apple App Store. Within the 170 allowed characters you can promote limited-time sales or inform users about new upcoming features of your app. If used, the promotional text will be shown on top of the long description. In contrast to the other elements on your Apple App Store page the promotional text can be updated at anytime without having to publish a new version of your app.

Keywords field (iOS only)

The keywords field only exists in the Apple App Store, and is invisible to store visitors and competitors. The keywords field can contain up to 100 characters per localization which should be separated by a comma only. Using additional spaces is not necessary and will waste your character allowance. The keywords field should contain relevant keywords that are not part of your app title or subtitle.

Category

Choosing the right category for your app to be listed in is crucial. It will influence how users can discover your app while browsing through the app stores. The position of your app in the category ranking depends on the number of downloads and uninstalls, as well as your ratings. The higher the number of downloads and the better the star-ranking for your app, the higher you will be listed in the chosen category. On the Apple App Store you can choose 2 categories, a primary and a secondary category. You will only be listed in your primary category. The secondary category only serves as a keyword, e.g. picking the "navigation" category for the secondary will improve your ranking for the keyword "navigation". On the

Google Play Store you can only pick one category. Keep an eye out for new categories, this does not happen frequently, but do not let it catch you by surprise! By being one of the first in a new category you can benefit from less competition. You can check all available categories for the Google Play Store[6] and for the Apple App Store online[7].

Optimizing the Visuals

Optimizing the visual elements such as app icon, preview video, and screenshots will draw the attention of potential users towards your app which will eventually have a positive influence on the conversion rate and therefore on the number of downloads. Furthermore, a clean design of your app store page will enhance the first impression a user gets of your app and set the tone for further interaction.

App Icon

The app icon is the very first visual element a user sees when finding your app in the app stores. Having a great icon does not necessarily guarantee tons of downloads, but it might help to get users to explore your app in more depth. A clean and eye-catching app store icon that is recognizable and under-standable will make your app stand out from the crowd.

Both stores require the icon to have a size of 512px by 512px. More detailed information on how to design your app icon can be found online under:

6   support.google.com/googleplay/android-developer/answer/113475?hl=en
7   developer.apple.com/app-store/categories

- Apple's App Store Connect: *help.apple.com/app-store-connect/#/dev8b5cb82e2* and Apple's Design Guidelines: *developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon*
- Google's Developer Portal: *developer.android.com/google-play/resources/icon-design-specifications* and Material Design Guidelines: *material.io/design/iconography/product-icons.html#design-principles*.

Generally, the icons in both stores should be cohesive in order to have a clear and consistent branding.

Screenshots

Screenshots allow you to comprehensively and visually communicate the best features and the most outstanding value propositions of your app. The screenshots are one of the most valuable elements on your app store page. Since visual information can be processed quickly by the human brain, potential new users are able to gain lots of information about your app by quickly looking at the screenshots. Make sure to provide them with relevant information and an appealing design to optimize your conversion rate.

There are numerous ways to conceptualize and design your screenshots. Viable approaches can range from a strict value and feature show-off, to a more creative approach that emphasizes on emotions and artistic aspects. The main point is, that potential new users should be attracted by your screenshots. The screenshots will support potential users by deciding if your app fits their needs and give them a first feel for your app.

Again there are some differences when it comes to the two app stores which should be considered. On the Apple App Store, one screenshot per localization is required, but up to ten screenshots per device (iPhone, iPad, Apple Watch, tvOS)

can be published. iPhone screenshots are required for 6.5-inch iPhone Xs Max and 5.5-inch devices; for iPads, 12.9-inch iPad Pro (2nd gen) and 12.9-inch iPad Pro (3rd gen) screenshots are needed. These screenshots will be scaled down for smaller device sizes. The first three screenshots will be displayed on the search results page and should therefore be highly informative and appealing. Further information on the different specifications can be found in the App Store Connect help section[8].

On the Google Play Store, up to eight screenshots per device (Smartphone and Tablet) can be uploaded. A minimum of two screenshots has to be provided in order to publish a store listing.

More in-depth information and specifications can be found in the Play Console Help[9].

Video

A video gives users a much more dynamic view into your app, with the help of animations, sounds, music, and voice captions. It is not required from the app stores but is recommended as a fun and engaging way to present your app to potential users. The video will be positioned in front of the screenshots and is therefore highly visible.

On Apple's App Store, up to three videos can be uploaded, but usually, one video should be sufficient. It will be shown within the search results page and autoplay directly by default. Therefore, having a video on the Apple App Store is crucial, since it will be a moving element in the search results page that will catch the attention of the user and increase the visibility of your app. The length of the video should be between 15-30 seconds and only in-app-footage is supposed to

---

[8]   help.apple.com/app-store-connect/#/dev910472ff2

[9]   support.google.com/googleplay/android-developer/answer/1078870

be shown. You are free to add music or a speaker to your video but keep in mind, that some users watch the video in silent mode, so make sure that they understand the benefits of your app without sound. In order to include a video, you will have to upload it to your App Store Connect Console. You can find some more information regarding the specification in the App Store Connect Help[10].

Google's Play Store, only allows one video to be added to your store listing. In order to do so, your video has to be published to YouTube first. Once this is done you can upload it to the Google Play Store by inserting a YouTube URL in the "Promo video" field in your Google Play Console. In comparison to the video in the Apple App Store, you are not limited to in-app-content when it comes to your video. You can reuse existing video material which explains the functions of your app, add an explanation text or even contextual scenes if needed. Try to stay within 30 seconds to keep the attention of the user. The video will be displayed on your app page before the screenshots. In order to watch the video users will have to click on the play button on top of the feature graphic, and then they will be taken to YouTube to watch it. Additional information about video specifications can be found on Google's developer page[11].

Feature graphic (Android only)

The feature graphic and the promo video are intertwined assets, in the sense that the feature graphic is required, but will not be shown on your app page without the video. When using a video in your app store listing in the Google Play Store the feature graphic is displayed before your screenshots. A

---

10 help.apple.com/app-store-connect/#/dev4e413fcb8

11 support.google.com/googleplay/android-developer/answer/1078870

Play button will be overlaid on the feature graphic, and users can watch the video by clicking on it. Consider the placement of the play button when designing your feature graphic and ensure a nice ensemble between the feature graphic and the following screenshots. Furthermore, animate the video so that the transition from the "thumbnail" that is the feature graphic to the first few seconds of the video is smooth so you do not irritate the viewers. You should also always use large font sizes when adding text here.

Google requires this asset as a 1024px by 500px JPEG or 24-bit PNG (no alpha).

Learn more

This chapter covered the basics of ASO which is an ever-evolving field. It is a complex topic and it is impossible to cover everything within one chapter. Apple and Google are regularly performing changes on their respective app stores which can influence the requirements. Therefore, it is important to frequently review the app store presence of your app and keep an eye out for changes. We hope this chapter has aroused your interest to learn more about ASO.

Here is a list of some elements which were not touched in this chapter but are still relevant for ASO:

— Release Notes/ What's New
— App Store Features
— Optimizing ratings and reviews
— ASO Analytics

If you want to learn more, search for those terms online and read on!

Christian Eckhardt

BY

User Acquisition and Retargeting

Users are oxygen for your mobile app. Regardless of how you intend to make a living from your mobile business, you will have a tough time making any money from it without having a healthy user base that continually uses your app (even more so if you tap into virtually any advertising-based business models). In this chapter, we will look at the basic mechanics behind acquiring and retaining users for your mobile application as well as the different activities that are part of it.

The contents of this chapter have a significant overlap with what would be called performance marketing in a lot of online/mobile-first companies. Without wanting to go into too much marketing theory here, performance marketing typically describes all activities that have the goal of acquiring & retaining users with the help of quantifiable marketing channels (so not Out-of-Home or print for example). If you want to put it like this, the opposite of performance marketing is the more traditional brand marketing where you will sometimes even use the same channels as in performance marketing, but the outcomes of your activities generally are very hard (or downright impossible) to measure. Having said that, there is still a significant amount of dependencies and interferences between performance marketing and brand marketing[1]. In most larger companies, these two fields of marketing are handled by separate (but ideally well-connected) teams.

It is not a secret that performance marketing means big business these days (particularly the user acquisition part)

---

[1]   thenext-us.com/2017/10/brand-marketing-vs-performance-marketing

with numerous participants in the mobile marketing value chain all trying to get their small (or not so small) piece of the big advertising cake. According to mobile attribution provider AppsFlyer, a whopping $38.9 billion have been spent in 2018 globally just on mobile app install campaigns[2] while mobile marketing is making up for around 70% of digital ad revenue already today[3]. With ludicrous YoY growth rates of over 20%, there is no end in sight for the growth of mobile advertising revenues[4]. Mobile has become the norm in the advertising world.

Interestingly enough there are not just a few app publishers out there where the entire business growth (story) is fundamentally based on performance marketing activities, meaning the number of users they have today is primarily made up out of users that were generated from putting money into marketing campaigns (so-called paid users). Typical characteristics of such companies are a strong online-based product as well as significant venture capital at hand (to spend on those marketing campaigns).

2   appsflyer.com/blog/app-install-ad-spend-predictions-2017-2020/

3   EMarketer: "Mobile makes up 70% of retailers' digital ad spend", bit. ly/2tNkhys

4   "Mobile Advertising Will Drive 75% Of All Digital Ad Spend In 2018: Here's What's Changing", bit.ly/33C4liL

User Acquisition

User acquisition describes all activities that aim to acquire new users for your mobile application. It is a vast field and often consists of online marketing activities for businesses that live in the online world (naturally).

From a processual perspective, user acquisition is about all the steps involved from a user seeing an ad for your product to clicking on it, being redirected to a corresponding website or the app store page and then installing your mobile application. The channels in which you can find new users can be categorized into paid media and owned media, as we will learn below.

In almost all companies that engage in user acquisition activities, the marketing manager on duty will make use of different available media channels/sources to acquire qualitative new users (forming what is called the media mix).

Paid Media

Virtually any marketing campaign where you, either directly or indirectly, pay money to the owner of the ad inventory (that displays the ad for your product or service) falls into the category of paid media marketing campaigns. There are some common examples of companies that only exist because they offer paid media campaigns (say Google and Facebook), generating the vast majority of their overall revenue in advertising, north of 80% to be more precise[5].

Paid media campaigns emit an almost magical appeal to marketing managers as they all come with the same highly attractive yet easy to understand value proposition: You insert money on one side and receive new users for your app on the other. Now the reality is not that simple, but the mechanisms

---

[5] "How Google Makes Money", bit.ly/2CY5oLr

behind all paid media campaigns follow the same pattern. You pay someone for a new user achieving some goal that was agreed upon (in our context this is typically a click on the ad, an app install or even some action inside of your app to be triggered by the newly acquired user). The payment between you as the advertiser and the owner of the ad inventory for the billable action you agreed upon, can happen either directly (in case of players like Google or Facebook) or somewhat diffusely via numerous intermediaries (smaller app publishers do not have the resources to market the little ad inventory they own, so they have to revert to ad-networks or so-called mediation platforms to sell their ad inventory).

Goals & Metrics

The apparent goal of any paid media user acquisition campaign is to generate qualitative new users for your mobile app. Metrics that are commonly used for such campaigns have naturally evolved around that goal and have been influenced by the needs of advertisers as well as the owners of the ad inventory (which naturally have slightly different intentions sometimes). The most important core metrics for you to know as you tap into the land of paid user acquisition are:

METRIC	DESCRIPTION
Impressions	An ad being exposed/displayed to a single user
Clicks	A user clicking on an ad
Installs	A user installing an app
Actions (Events)	A user performing some kind of action in your app

While the most popular acronyms (largely resulting from the core metrics above) for you to know in regards to paid user acquisition are:

METRIC	EXPLICATION	DESCRIPTION
CPM	Cost Per Mile	Very traditional online marketing metric that describes the price of one thousand ad impressions.
CTR	Click-Through Rate	The percentage obtained from dividing the number of users that clicked on a given ad by the number of times the ad was delivered (=impressions). Important indicator for the quality of the creatives that you are using (the higher, the better).
CPC	Cost Per Click	The price for one user clicking on your ad.
CPI	Cost Per Install	The price for one user installing your mobile application after clicking on your ad.
CPA	Cost Per Action	The price for one user performing some predefined action inside of your mobile application. Sometimes also referred to as CPE (Cost Per Event).
CLTV	Customer Lifetime Value	The predicted or calculated net profit that you are going to make from the relationship with a given user.

METRIC	EXPLICATION	DESCRIPTION
ROI	Return on Investment	The percentage obtained from dividing the net profit generated by the marketing campaign by the net spend for the marketing campaign times 100. In contrast to ROAS, this metric takes into account all expenditures that were invested into the campaign (software, labor, costs,...) which means it is harder to achieve a positive ROI than it is to achieve a positive ROAS.
ROAS	Return on Advertising Spend	The percentage obtained from dividing the revenue made from ads by the advertising spend times 100. In contrast to the ROI, this metric only takes into account ad spend and revenue made from the acquired users. Essentially ROAS is only considering revenue while ROI is considering profit.

Selected Channels

The channels to reach qualitative new users for your mobile app are ever-evolving (as is the way people use their smartphones). Generally speaking, you can only reach users effectively in other apps where they spend a significant amount of time. Obviously, those apps have to offer ad inventory so that you can run your ads there. The four quintessential meta-categories of channels for paid user acquisition are:

Paid Social

Advertising is the primary revenue stream for virtually any of the social networks these days. Facebook, Twitter, Snapchat, and many more will be happy to take your precious marketing budget to run ads for your product. Because these platforms

appear as such an obvious choice for marketers looking to engage with new users, competition with other advertisers (and hence prices for ad inventory) is continuously on the rise for any of these. Therefore, it is essential to efficiently target and optimize one's campaigns in order to stand out of the flood of marketing campaigns, which users are confronted with every day.

Billing for the ads you run on these platforms typically happens on models that are related to either the impression (CPM), click (CPC) or install (CPI) while some even offer models that are based on users taking some specific action in your app after the install (cost per action or CPA). Almost all advertisers that are serious about their online marketing, engage in some sort of paid social activities.

Paid Search

Google has always been and still is the pioneer in this area, but with Apple rolling out Apple Search Ads[6] to more and more storefronts globally, there finally is the option to run ads on the search function of the App Store as well. In general, paid search describes all occasions where you can display ads after a user has been searching for some specific keyword(s). The two main use cases for this in the app context are the search on the web (where Google is still ace) as well as search queries in the Google Play and Apple App Store. For the latter, you have to use Google Universal App Campaigns (UAC) on Android[7] and Apple Search Ads on iOS to be displayed above the organic search results (and hence your competitors) in the app stores.

---

6   searchads.apple.com

7   developers.google.com/adwords/api/docs/guides/mobile-app-campaigns

Again, billing for the ads you run on these platforms typically happens on models that are related to either the impression (CPM), click (CPC) or install (CPI) while some also offer user-action (CPA) based models.

Ad-Networks

Traditional ad-networks are directly connected to many publishing apps and try to sell their traffic on a semi-exclusive basis. Unfortunately, they represent the most intransparent traffic as they will typically not disclose the actual publisher apps to you (as this is their only asset really). This diffuseness makes this kind of business model highly attractive for fraudsters that apply all types of techniques to fake results (regardless whether it is ad clicks, installs or even in-app activity). Billing for the ads you run on these platforms usually happens on models that are related to the app install (CPI) while some might be willing to offer you CPA-based models.

Given the limited transparency of such offerings, you should treat any campaigns, that you run with such networks, with extra caution and question the numbers you get out of them (especially if they offer you CPA-based billing).

Real-Time-Bidding (RTB)

A relatively new trend when it comes to user acquisition is the rise of RTB (Real-Time-Bidding) platforms as well as providers who specialize in acquiring app installs through the big SSPs (Supply Side Platforms). What they do is placing bids for ad inventory at so-called ad exchanges. This allows them to buy traffic from a multitude of different apps in an automated manner. The various vendors in this area compete against each other by using their own (supposedly superior) proprietary algorithms, as well as machine learning to find the right users to place a bid on the auctions. Billing for the ads you run on these platforms typically happens on models that are related to either the impression (CPM) or the click (CPC). Providers worth

taking a look at here are *Jampp.com* and *Liftoff.io* (amongst countless others).

Tracking

To be able to run paid user acquisition campaigns that are based on anything else than the mere amount of impressions and clicks, you have to integrate additional tracking solutions into your mobile app. The reason for this is that 3rd parties (like Google or Facebook) have a tough time figuring out whether an install of your app has happened after a click on one of their ads without such additional tracking solution. They can not fetch this information from the Google Play and Apple App Store, meaning that in consequence, they can also not sell you app installs (as they are unable to detect that an install has happened). That is why it is no surprise that the most basic install tracking solutions come from someone that has a natural interest in enabling the measurement of app installs and user behavior based events for your app - the advertising platforms themselves. Facebook and many of the more significant advertising platforms offer their own proprietary tracking solutions so that you can run advertising campaigns on their platforms. The two biggest problems with these: They only work for measuring campaign results on the advertising platform that is handing them out (creating a similar situation like you have on web where every advertising partner forces you to integrate a proprietary tracking pixel on your website to run campaigns with them) and secondly, it is just not advisable to receive the measurement of the campaign results from the same party that is selling you the traffic.

Instead, it is recommended to integrate a 3rd party install attribution provider (they call themselves MMPs - Mobile Measurement Partners) that can independently attribute new app installs to the source (e.g., your marketing campaign on a

certain website). There are four relevant vendors in the market for this today: *Adjust.com*, *AppsFlyer.com*, *Branch.io* & *Kochava.com*. Each of them is working in more or less the same way. They collect information from the user upon clicking on the ad and save it on their servers. Then there is a component of the tool that you have to integrate into your app (a so-called SDK - Software Development Kit). Once implemented, this SDK runs inside of your app and collects information from the user as well. When the user installs and then opens your app for the first time (after clicking on the ad of course) these tool providers match that two information (click & install) and attribute the install to the source. After that has happened, they would then also forward this information to the advertising platform which will then be able to count the install (and subsequently bill you for it). In addition to the install tracking, the SDK of the MMP would also be able to measure certain activities that your users perform in your app (which you can freely define or even choose not to capture entirely). This behavioral data can be forwarded to the advertising platforms as well and will then be available to optimize your campaigns (i.e., buy more users that perform the desired actions in your app). Ever wondered why that pair of socks from Amazon you looked at three weeks ago is still chasing you in different ads across different mobile apps you are using? This is how they do it.
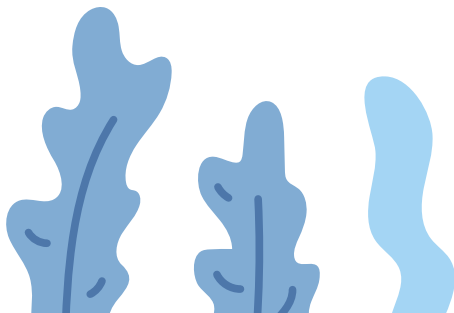
Owned Media

While all the paid marketing activities for user acquisition described in this chapter so far might seem attractive to you, they all have one major drawback: They cost money (typically a scarce resource if you are just starting out with your mobile business). That is why we will now speak about a kind of placement that is not costing you any money - your very own ad inventory.

If you have a website or some other kind of online presence that goes along your mobile application, chances are that you have unused potential to convert some of the users that visit your website/blog/social media page to actual app users. As paid marketing has become the norm and cure for everything, the potentials of owned media placements are often disregarded. The general idea behind all the placements listed below is to show the visitor that you have a mobile app that is compatible with his device and that he or she should certainly try it out.

Goals and Metrics

The goal behind any owned media user acquisition activity only differs slightly from the objectives behind paid user acquisition campaigns. Similarly, you try to generate new users for your app - the only difference being, that now the users are already on your website and as you own the inventory, you do not have to pay to show your ads.

Since there are no costs attached to the ad placements themselves, we are only looking at a reduced subset of core metrics, which should be taken into account for your owned media user acquisition activities, compared to the paid user acquisition activities:

- Clicks
- Installs
- Actions (Events)

While the most popular acronyms (largely resulting from the core metrics above) for you to know in regards to owned media user acquisition are:

- CTR (Click-Through Rate)
- CLTV (Customer Lifetime Value)

Example Placements

Possible options for placements in your online presences are numerous and depend heavily on the kind of service or product you offer. The three most effective and most universal ones are:

Smart Banner

Both Android and iOS offer standard versions of the smart banner, which hovers on top of your website and for visitors using a smartphone leading directly to your app's store entry. If you use a tracking provider (MMP) in your app, it is advisable to create a custom smart banner and put a tracking link behind it, so that you can measure how many installs you generate from it (+ some extra information).

App Store/Play Store Badges

It is advisable to put the official Apple/Google app store badges (standardized buttons opening your app's store entry) somewhere at the bottom of your website if your app relates to the content of the website. Doing so gives your users another option to get your app and shows that you have one in the first place.

Check Apple's and Google's developer pages to download the assets and learn more about the respective guidelines on their usage:

— *developer.apple.com/app-store/marketing/guidelines*
— *play.google.com/intl/en_us/badges*

QR Code

For website visitors using a regular computer, it is helpful also to include a QR-code to the App Store/Play Store so that they can take out their smartphone and scan the code, instead of having to look for your app on the stores. If you use an install attribution provider (MMP), this exercise becomes a bit easier because they all offer tracking links that can determine the correct app store (Android/iOS) for the user to be redirected to based on the user's device.
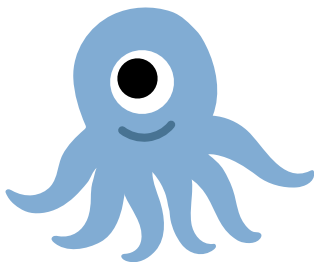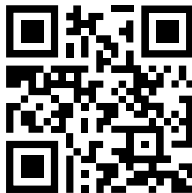
As you can probably tell by now, it makes your life a lot easier if you use an attribution tracking provider (MMP) when implementing the owned media placements mentioned above. Indeed, it is not an absolute necessity to have a tracking provider in place (in contrast to paid media), but if you do, you should use tracking links behind all the placements so that you can get the clicks, installs, and events resulting from your owned media activities.

Furthermore, it is advisable to create a small script that fetches the so-called referrer URL and any incoming UTM parameters coming with the website where the user clicks on your tracking link. This information should then be written into the tracking link you use on the page as tracking parameters (right in the moment when the page is loaded). This way, you can see where visitors from your website are coming from before installing your app in your MMP dashboard/reporting.

Retargeting

Acquiring new users for your mobile app is only half of the story (actually less than that). In order to establish any sustainable app business, you need loyal customers that come back to your app over and over again. All the precious installs you can buy with marketing money will not take you anywhere, if none of these paid users show any activity after the install (which is often the case with low quality mobile traffic). The technical expression for users becoming inactive, is user churn - if you start reading more about this, you will most certainly come across the term in a lot of publications as the problem is describes as somewhat trending these days (and even more so the available countermeasures at hand).

The sheer figures are quite depressing. According to a study from mobile legend Andrew Chen dating back to 2015[8], around 80% of new users are gone just three days after they installed the app on their Android smartphone. That is they either stopped using the app or just downright uninstalled it from their device.

Average retention curve for android apps
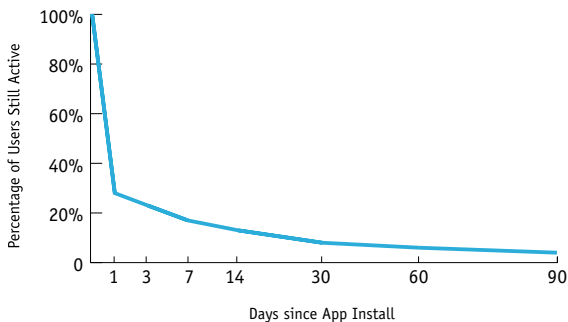
These figures were confirmed with more recent data throughout the following years, leaving a rather dismal image of how users engage with the apps on their smartphone.

If you study the problem of user churn for mobile apps on a meta-level, you end up with three fundamental solutions (or countermeasures) that should always be seen as complementary to each other, rather than alternatively.

---

[8]   "New data shows losing 80% of mobile users is normal, and why the best apps do better", bit.ly/2AjOvMS

A) Paid Retargeting

You run paid ads in other apps/websites exclusively targeting existing app users of yours that are not as active in your app as you would wish (because they either abandoned your app or entirely uninstalled it from their device). By showing them additional features of your product or sometimes just raising awareness that the app already exists on the user's phone, you try to convince them to get back into your app and to give your product another shot. Due to the level of technological sophistication involved in the process of buying ad-inventory in a real-time manner, you have to rely on one of the dedicated mobile app retargeting providers in the market such as Remerge[9].

B) mCRM (Mobile Customer Relationship Management)

Think of mCRM as the owned media counterpart of paid retargeting. You make use of communication channels that do not incur any (or just minimal) incremental costs per message sent like push notifications, in-app-notifications, email and so on, again trying to convince the user that he should come back to your app. The obvious downside is that with the majority of messaging channels that are coming with your app directly (namely push notifications and in-app-notifications), you will not be able to reach any users that have already uninstalled your app. If you want to succeed in the mCRM domain, it is almost inevitable that you end up using a technology provider that handles user segmentation, message delivery, campaign triggering, and reporting for you. The topic of mCRM will be portrayed in greater detail in the next major chapter of this book.

---

9   remerge.io

C) You build the perfect product

In theory, all the market shares should be yours if you "simply" go out there and craft the ideal app. While that scenario is never going to happen, it is a matter of fact that a higher quality product will almost always have better customer retention rates than a similar product of lower quality.

As you will have noticed, only two of those countermeasures even fall into the marketing category (paid retargeting & mCRM). Consequently, paid retargeting will be discussed in this sub-chapter, while the other marketing-based cure for user churn (mCRM) will be the main actor in the next major chapter.

Paid Retargeting

The character of paid retargeting in itself is similar to paid user acquisition, only for users that already have (or have had) your mobile application installed on their device. The main goal is to raise additional awareness for your app by displaying ads to existing users in order to get them back into your app. Thanks to so-called deep-links, the user experience can be designed pretty smoothly. For example, you can decide to take users right back into your app, after clicking on the ad (vs. having them to go through the app store again). Typically you would only target users that are not living up to your particular definition of a loyal customer with this kind of campaign and then try to convince those people to give your product another go. The obvious downside is that you have to pay someone a good amount of marketing money again to go out, find the users that already have your app while they browse other apps and websites (utilizing so-called user lists with device identifiers of the users you already know), buy an ad impression from this specific app publisher and eventually display a banner or some video to the user you are retargeting. All of this has to happen in a staggeringly short period of time

(the time between a user clicking on a link to a new page/ screen in the app and the desired page having finished to load - speaking milliseconds here) which is why you will have to use one of the RTB (real-time-bidding) providers.

Goals and Metrics

Paid retargeting campaigns are typically created with a clearly defined goal in mind: Some action or behavior that users should perform in your app. Think of app retargeting campaigns as a tool to push users a little bit further down the conversion funnel. They have already installed your app but did not place an order yet or did not buy that extra pack of in-app-currency you offered them two weeks ago. With retargeting ads, you would now be able to remind them of your app/service and make them come back to your app to perform the desired action eventually.

Since users are already in the later stages of the conversion funnel (where install has already happened) we are looking at a smaller subset of important core metrics here:

— Clicks
— Actions (Events)

While the most popular acronyms (largely resulting from the core metrics above) for you to know in regards to paid user retargeting are:

— CPA (Click per Action)
— CLTV (Customer Lifetime Value)
— ROI (Return on Investment)
— ROAS (Return on Advertising Spend)

Methodology and Vendors

There are numerous vendors for app retargeting campaigns offering their supposedly superior campaign solutions to solvent app publishers wanting to re-engage their user base. However, from a technological and processual perspective, the different offerings do not actually differ too much. The only real differences are in the quality of the algorithms, which they use to place bids for ad inventory on the big ad exchanges, leading to better efficiency for some of the vendors.

Outside of the bidding algorithm, they all work in the same way: When starting the campaign, you provide them with a list of users you want to retarget. Users on that list are typically identified by the GPS_ADID for Android and the IDFA for iOS (being the two dedicated advertising identifiers from Apple and Google). Secondly, you have to define the creatives, the goal of the campaign (what should users do in your app?) and set up the data transmission from your MMP so that the event data is passed to the retargeting provider. When all the preparational work is done, the retargeting provider starts to place bids for ad inventory being displayed to users on your user list at the big ad exchanges like *MoPub.com* or *PubNative.net*. Those exchanges run auctions for ad inventory right at the moment when a page on a website or inside of an app is loaded. At that time, the information about what user is going to see the ad (based on advertising ID) is already present and the retargeting providers would then only place bids for users that are on your user list. If they win the auction, your ad is displayed, and ideally, the user would click on it. Upon the click on the ad (which has your MMP tracking link behind it), the user gets taken back to your app straightaway while the MMP SDK would track all the user behavior after

the app opens again. Events that occur are forwarded to the retargeting provider so that they can optimize the campaign based on this.

Having made the point of how similar the providers are to each other, there is still a hand full of vendors that we can recommend from personal experience and based on the technology in use. These are *Jampp.com*, *Liftoff.io*, *Remerge.io* and *Adikteev.com*.

Tracking

Having an MMP in place is even more critical for retargeting campaigns than it is for user acquisition campaigns. It is downright impossible to run any retargeting campaigns without an MMP because you will lack the most quintessential part of the retargeting campaign setup: The user list that the retargeting provider needs to look for your existing users as they use other apps and websites. This list is maintained by your MMP (as the SDK runs in your app, capturing all your current users).

Despite the user list, your MMP plays another vital role in executing retargeting campaigns by measuring user events (actions your users perform in your app). One of these events will most certainly be the goal of your retargeting campaign and without your MMP tracking it in the first place and forwarding that data to the retargeting provider you simply can not run the campaign (i.e., the retargeting provider would refuse even to start the campaign as there is no success criterium).

Deeplinking

From a user experience perspective, it is (very) advisable to use deep-links when setting up any retargeting campaign. In combination with the mechanics of the tracking link you get from your MMP, this allows you to redirect the user right back

into your app upon clicking on the retargeting ad (vs. having them go through the app store again). Ideally, you would redirect the user to the view/page in your app that corresponds with the content of the ad that the user clicked on in the first place. To give you an example: If you run a fashion online store app and you are displaying retargeting ads showing sneakers, make sure that the user gets the correct deep-link to be redirected to that particular pair of sneakers (and not have him land on the homepage of your shop).

If you reach the point where you want to run retargeting ads for your app, and you do not support deep-links by then, it is high time to get them implemented. There is no extra tool needed to achieve this. You only have to declare the deep-links and corresponding views in the IDE (integrated development environment) that you are using. Both Apple[10] and Google[11] provide detailed instructions on how this can be done.

10   apple.co/2Ki9JAg

11   developer.android.com/training/app-links/deep-linking
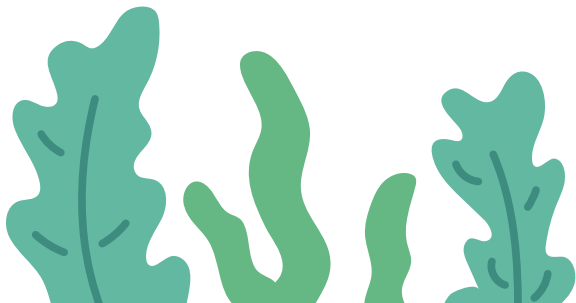
Mobile Customer Relationship Management (mCRM)

Generating new users for your mobile app is only one side of the medal. If you set out to establish any kind of (successful) long-term app business, you need to build a robust foundation of loyal users that utilize your app continually and over a prolonged period. After we have been looking at paid retargeting activities (i.e., showing ads to existing app users of yours, inside of other apps or websites) as a counteractive measure in the last chapter, we will now move on to another popular countermeasure to user churn: Mobile Customer Relationship Management (mCRM)

Compared to paid retargeting, you can think of mCRM as the owned media counterpart, so to speak. The general approach in mCRM is to make use of communication channels that come with your app (e.g., push notifications or in-app messages) or are rather easy to get into such as e-mail, trying to convince the user that he should come back to your app. In contrast to paid retargeting where you have to pay for every ad impression that is served to a user, mCRM relies on channels that typically do not incur any (or just minimal) incremental costs per message sent (say push notifications). Generally speaking, there are development resources (and hence costs) attached to setting up a mCRM system in almost all cases though. Typical content that you would deliver to your users as part of your mCRM activities would be new product improvements, special promotions, relevant external events (if applicable) or even just a general introduction to some features that the customer

was not using until this point. If you want to succeed in the mCRM domain, it is almost inevitable that you use a technology provider that handles user segmentation, message delivery, campaign triggering, and reporting for you.

With CRM in itself being a traditional marketing discipline, it is fascinating to see what kind of symbioses it forms with the actual product experience in the mobile app realm. It is impossible to draw a clear line between what is still considered as the product user experience (typically dictated by product managers and UX designers) and what is already mCRM (managed by the marketing managers). When done right, your users will not be able to tell that the friendly onboarding they are receiving is already the very first part of your set of mCRM campaigns. Regardless of these more theoretical questions, it was proven over and over again that the best mCRM campaign experiences are crafted when the marketing teams work very closely with the responsible product teams (or are represented by the same person(s) even).

Communication Channels

There is a large variety of messaging channels that can be utilized for mCRM activities, reaching from the usual suspects like push notifications to corner cases like SMS (in most countries) or even physical mailing. Depicting all of them would go way beyond the scope of this chapter, meaning we will focus on the most popular and most widely accepted ones for now.

It helps to look at the different messaging channels in a matrix between urgency (how urgently do I want to have this communication being delivered to my customers) and content richness (how much content does the message contain). This

concept was first brought up by mCRM tool vendor Braze[1] and subsequently found quite a bit of public interest in the mCRM field.
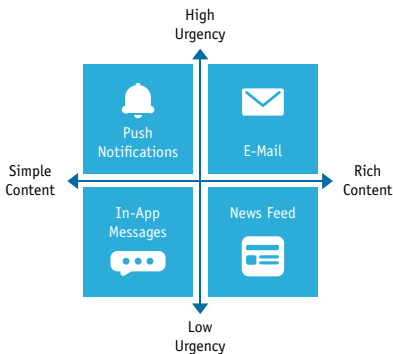
In the illustration, you can see the four most extensively practiced messaging channels in mCRM and how you would use them depending on the nature of your messaging. This way of thinking goes back to one of the core concepts of mCRM in general: You would always plan the content and goal of your message first and then look for the adequate messaging channel - not the other way around.

A second general rule of thumb in mCRM is to always start with the least intrusive messaging channel for whatever communication you want to deliver and then to only gradually increase the obtrusiveness by selecting one of the more intimidating messaging channels as required by external circumstances (i.e., you would send somebody an e-mail with your latest summer sale first, and only if the user does not

---

1   braze.com

open the e-mail, you would maybe want to think about sending a reminder as a push notification a few days later - not the other way around).

Let us take a look at those four most popular messaging channels in greater detail and discuss some of the more uncommon cases right after that.

Push Notifications

Traditional push notifications are by far the most widely adopted means amongst all the mCRM messaging types whatsoever. Everyone that has ever used a smartphone (i.e., the majority of the population) is familiar with the little notifications delivered directly to the home screen via the operating system, and for most users, it is a bit of a love-hate relationship these days. Marketers have realized pretty early on, that push notifications are a very direct (and hence fairly intimidating) way of communicating messages to their users, despite both Apple and Google (in their respective developer guidelines) explicitly prohibiting push notifications that can unmistakably be identified as advertising or possess an overly commercial character. Push notifications have been the subject of ongoing debates in the past and what is socially acceptable with your users, in particular, depends heavily on the geography (e.g., push notifications with marketing content are more accepted or at least tolerated in Latin America while the acceptance threshold for most European users will be much lower) as well as other factors like your app vertical.

If you manage to deliver relevant and helpful content to your users that does not annoy them in their everyday life, though, push notifications can be a highly potent communication tool between you and your users. This golden rule applies to all messaging channels we are looking at here but push notifications are clearly the type of message that is the most

prone to annoy users, given that it is delivered so directly to the screen of the device and competes with a multitude of other notifications there (supposedly from family & friends because of all the messenger apps). On that account, it is no surprise that opt-in rates for push notifications (which users have to grant in your app using a predefined permission prompt before you can send them proper push notifications, at least on iOS) will be meager if you do not manage to deliver meaningful content to the user. You are all set up for a terrible start already if you do not give the user clear and reasonable arguments as to why he should be allowing you to send push notifications in the first place (which you should display prior to the default system-side permission prompt). In addition to that, the user can opt-out from push notifications in the system settings on both Android and iOS at any time. The opt-out situation also symbolizes the most significant downside about traditional push notifications as a channel for mCRM - if you lose (or never get) the user's consent, you will not be able to send him any push notification whatsoever.
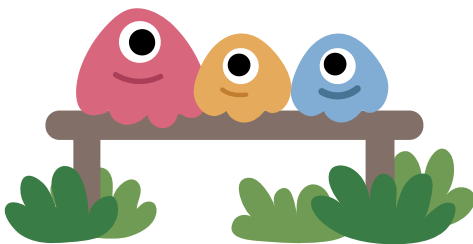
In-App Messages

Another pretty popular messaging channel for mCRM is the utilization of so-called in-app messages. It might look like this:
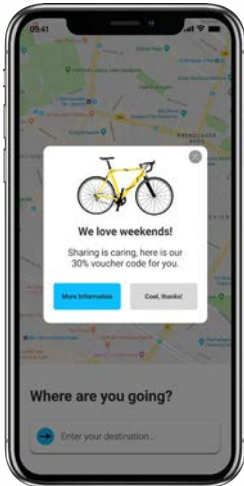
They come in many different shapes and sizes (from interstitial to full screen to floating at the bottom/top of the screen), meaning that you have pretty much endless possibilities in designing and testing them. What they all have in common: They are displayed inside of the app, which means they are only visible for users that are already using your app (ruling them out for virtually any campaign that is aiming to bring users back into the app). The real beauty of in-app messages lays in the fact that the marketer can create and run them using one of the mCRM tools we will look at below, without having to bother the developers to touch the code of the app again (let alone having to wait for the next app release and until users have adopted the new version).

Very often in-app messages are used alongside regular push notifications to present some additional information to the user inside of the app after he clicked on the push notification (a critical aspect of providing meaningful content to the users).

News Feed

Think of a blog (with any information you as a marketer want it to carry) that lives right inside of the app. This is the so-called news feed (sometimes also referred to as the app inbox).

The news feed is a functionality that (the better) mCRM tools in the market offer today, and that is integrated into the app once (same as with the in-app messages). Typically, the news feed would live somewhere in the side menu as a news or updates section (to be decided upon integration). Again the significant advantage is that the marketer can deliver content to the app without having to touch the code or even ask the developers. In contrast to the in-app messages, the content of the news feed is even more persistent, meaning you can put information there which users might refer to again at a later point (say voucher codes).

E-Mail

Measured by opens, the overall e-mail messaging channel is dominated by mobile devices today. Depending on which figures you look at, the point at which more e-mails were opened on mobile devices than on desktop computers has

been reached somewhere between 2015 and 2017[2]. You should design any e-mail that you compose accordingly. In particular, this means a mobile-friendly design as well as using (universal) deep links wherever possible to redirect users straight back into your app when they click on one of the links. If you have an attribution provider (so-called MMP, see previous chapter) integrated with your app, it is advisable to use tracking links with appended deep links in your e-mails to track what users are doing after they click on any of the links in your e-mail.

From an mCRM campaigning perspective, it is vital to include e-mail as a channel for reaching mobile app users in your planning so that the messaging they receive, feels in line with what they get on the other mobile channels (push notifications, and so on).

Edge Cases (SMS, Physical Mailing)

Numerous additional messaging channels can be utilized as a part of your mCRM strategy. For a telecommunication provider owning the associated infrastructure, SMS might make perfect sense as a messaging channel, for example. The same might hold true for physical mailing (i.e., letters) if your target group dictates that.

The general rule of thumb for all of these channels: The more uncommon they are, the more often you will not see them as default out-of-the-box offerings from the mCRM vendors. Rather than that, these additional channels will most likely be made available via some API from the mCRM vendor to a 3rd party tool that is much more focused on the specific channel such as Twilio for SMS[3]. In any case, you should make

---

2   "Mobile is now preferred platform for reading email with more than half of
     all email opens", bit.ly/2GZTma9

3   twilio.com

sure that the communication you send via these alternative channels is in line with what you present to your customers using your more traditional online mCRM channels and ensure that you track the results of the campaign in these other channels if possible.

Campaigns

All activities in the mCRM realm follow the same logical structure. You will always have a campaign that is being sent to a particular group of users (the user segment). A campaign can consist of different messages and can make use of different messaging channels (like push notification and e-mail) at the same time.

For every campaign to be sent, you need to define a user segment and at least one message to be contained in the campaign. What you want to receive once the campaign is completed (i.e., after the messages were sent) is the campaign performance data, showing you the results of your campaign. Data that you would expect to be included in this would be clicks on the push notification (if applicable), push notifications sent (if applicable), e-mail opens (if applicable), app opens, as well as particular actions that the user takes in your app after receiving the message (which you defined upfront as the goal of your campaign). While you can theoretically run mCRM campaigns without this campaign performance data, there is not much of a point in doing so, as it would be a complete stab in the dark with no way of optimizing anything, really.

In the next subchapters, we will take a closer look at all the components of mCRM activities mentioned above.

User Segmentation

It all starts with the user segmentation. Here you would define, which users you want to target with your campaign. Segmentation typically happens based on user attributes (like age, gender, location, and so on) as well as user behavior (usually tracked in the form of events like "placed an order", "signed up for the newsletter" or "completed registration form"). For most use cases, you will want to run the segmentation dynamically - meaning that the segment of users is not just a mere snapshot of all users that fall into the criteria when the segment is created. Rather than that, you want to have the defined requirements for the users to fall into the segment being monitored on an ongoing basis so that you have the most up to date list of recipients when the campaign trigger occurs (see next subchapter). The information that you base your user segmentation on, has to be tracked/captured for each user first though (typically via the event & user attribute tracking of your mCRM vendor).

Campaign Triggering

The campaign trigger, as the name already implies, is the actuator for your campaign to start sending the messages. The more ordinary campaign triggers evolve around time with the utmost basic trigger being the exact moment when you hit the "launch campaign" button. More sophisticated triggers are based on different events to happen, and they can also carry a delay if required (e.g., "send the user the onboarding push notification 2 hours after he first opened the app" or "remind the user of the item in his basket if he does not proceed to checkout within 30 minutes after adding it to the cart"). The information that you want to base your triggers on has to be tracked/captured for each user first though (typically via the event tracking capabilities of your mCRM vendor).

Messaging & Personalization

The messaging is the content of your mCRM activities. As such, it is mission-critical, and you should invest a good amount of consideration into what kind of content would be relevant for the user as well as what the exact caption should be for each message. The other question you have to answer here is which (combination of) channels you want to use, in order to transmit your message to the user. The last important aspect to mention in this area is the personalization. There are two points to it:

— Personalizing the kind of content you send to the user based on his/her behavior and preferences (take a look at what Netflix does for their apps with push notifications around recommendations and suggestions for content based on your previous behavior)

— Personalizing the actual caption using placeholders for supposedly personal information. If you send messages at scale, it becomes unfeasible to contact every user you got manually and to put his or her first name into the first line of the message (not that this would be recommended anyways). Instead, you should make use of smart placeholders that allow you to send messages at scale and in an automated fashion, while still appearing to be super personal for the user. The information that should be available in the placeholders has to be tracked/captured for each user first though (typically via the event & user attribute tracking of your mCRM vendor).

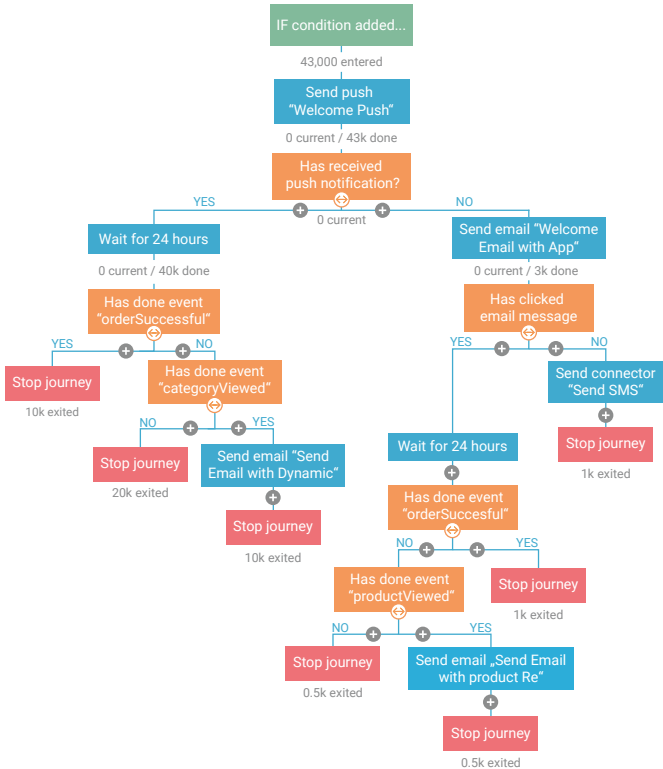### Campaign Performance Measurement

With the campaign performance measurement and reporting part, we have come full circle. This data will tell you how well your campaigns did, how many users clicked on your push notification, how many opened your e-mail, and so on. Your mCRM vendor typically captures all of this. With this information, you can make informed decisions to optimize your campaigns and potentially even trigger new campaigns, depending on whether the user opened that one e-mail or not. It is also what you will have to report if someone higher up in your organization asks you whether the uplift you are getting from your mCRM campaigns is really worth all the efforts of the integration (of the tool). On this occasion, it is also helpful if you have set up a global control group of users (we recommend ~ 10%) that never received any messaging from you to prove, that the ones who did are indeed generating more revenue than the control group.

Automation

As we have learned above, there are many moving parts involved in creating a sophisticated mCRM setup to enhance the experience of your app users. With all the mechanics, technical infrastructure, logical operators, tracking data and rule engines needed to run mCRM campaigns at scale, it is important to stress again that virtually any company that wants to be successful with their mCRM efforts, ends up using one of the established SaaS vendors in the space sooner or later. These tool vendors handle user segmentation, message delivery, (user) data capturing, campaign triggering, and reporting for you and all come with some sort of web-based GUI. It is no surprise that the development efforts that went into their solutions are far away from something that you might be able to

build yourself in-house as a little side project. Big corporates with significant development workforce have failed miserably trying to do that, and it is even more unrealistic for smaller companies just starting off to succeed at this adventure. It is the same story as with a lot of SaaS vendors and companies trying to replicate their functionalities in-house today: If you actually manage to build a reasonable mCRM solution for mobile apps as a side project, you will most certainly be better off pivoting your entire business into becoming the new rising star on the mCRM SaaS sky (making your side project the new main project vs. continuing your previous business).

One area that is especially hard to replicate from the features that modern mCRM tools have to offer is the automation piece. Interestingly enough, marketing automation is even used as a synonym for mCRM sometimes, because automating your campaigns is such an essential part of the whole mCRM game. What marketing automation describes is the condition of having all your different mCRM campaigns lined up neatly in a way that they would all be triggered automatically (and often depending on each other) based on user behavior and other external events, without any manual human input. This is extremely difficult to achieve, and most marketing managers already fail at even just visualizing all the mCRM campaigns they run and the interferences between them. The leading mCRM vendors can do the heavy lifting for you here, and all of them come with some kind of journey/flow builder that looks something like this:

IF condition added...

43,000 entered

Send push
"Welcome Push"

0 current / 43k done

Has received
push notification?

YES   0 current   NO

Wait for 24 hours

0 current / 40k done

Has done event
"orderSuccessful"

YES   NO

Stop journey

10k exited

Has done event
"categoryViewed"

NO   YES

Stop journey

20k exited

Send email "Send
Email with Dynamic"

Stop journey

10k exited

Send email "Welcome
Email with App"

0 current / 3k done

Has clicked
email message

YES   NO

Send connector
"Send SMS"

Stop journey

1k exited

Wait for 24 hours

Has done event
"orderSuccesful"

NO   YES

Stop journey

1k exited

Has done event
"productViewed"

NO   YES

Stop journey

0.5k exited

Send email „Send Email
with product Re"

Stop journey

0.5k exited

In this case, the example is borrowed from MoEngage[4], but other vendors have similar offerings. This example campaign flow starts with a user first opening the app after the install (i.e., the onboarding) and runs through different decision trees of possible campaigns, depending on user behavior and other factors like simple time delays. It shows you in one illustration why it is so incredibly difficult to build all those campaign logics and interferences between campaigns yourself.

If you want to learn more about the many-faceted topic of mCRM, you should certainly take a look at the vendors that are leading the pack at the moment:

— *Braze.com*
— *CleverTap.com*
— *Leanplum.com*
— *MoEngage.com*

4   moengage.com/flows

BY Linda Harnisch
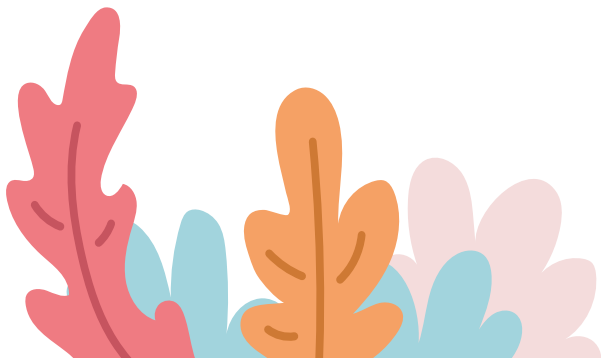
Mobile Analytics & User Feedback

If you are a mobile app marketer, product manager or app developer, would you trust your gut feeling in order to craft outstanding mobile app experiences for your users? Probably not. According to McKinsey, "companies that use customer analytics outdo their competition in terms of profit almost twice as often as companies that do not"[1]. In this chapter, we will focus on the value for app publishers to gain understanding beyond numerical data such as app downloads. But look at patterns in user behavior by means of quantitative and qualitative analytics to reach their business goals.

Getting users to download your app does not mean they will stick around. One of the biggest challenges and barriers around app audience growth is linked to user retention[2]. According to numbers by Localytics, over 70% of app users churn[3], due to app deletion, after 90 days. This is disappointing because app companies work hard and burn marketing cash on user acquisition (UA) campaigns. In order to retain existing users, you have to start collecting user behavior data inside your app. This is why you need mobile analytics.

Mobile analytics in this chapter refers to a method of collecting user behavior data in iOS or Android apps, gathering insights into the hidden lives of your app users and act on them. These data points guide marketing, sales, product

---

1   "Why customer analytics matter", mck.co/2KdAEuX

2   You can learn more about retaining app users in this book's chapter: Mobile Customer Relationship Management

3   info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate

and development teams in making data-informed decisions. Ultimately, you want to know how well your app works and what to optimize.

This chapter will answer questions around mobile app analytics that push beyond features and functionality of the traditional web. Familiar metrics and mechanisms from your web analytics experience might not get you far at all. We are just using mobile apps completely different than the web. In this chapter you will learn what type of analytics information is gathered; from quantitative data (performance, product, and marketing analytical data) to qualitative data insights via user feedback. By the end of this chapter, you will understand why everyone from marketing to product and development needs analytics data.

Defining Goals

Without hard data, app publishers and developers are left flying blind. Or they fall prey to arbitrary feature development sourced from their intuition or eager managers. Analytics data provides a base from which different app business units can actually strategize. This is powerful for marketers and developers alike and enables them to create an app experience that is more useful and engaging for their users. For instance, if an e-commerce app publisher discovers that a big amount of their users drops out in the shipment screen, there must be a technical problem or an issue with the user interface.

Before you immerse yourself in the mobile analytics world, it is worth understanding how to set the stage for your analytics strategy. This should be rooted in your Key Performance Indicators (KPIs) and the business value you hope to derive from the app.

Unless you are developing your app in a small start-up,

there are often many business units working alongside to develop, optimize and promote an app. Each of these departments - product, marketing, business development, the technical team and management - should make up their mind about:

— What they want to accomplish.
— What their key performance indicators (KPIs) are.
— How they will determine whether they are achieving them.

Establishing a KPI hierarchy for all departments helps to counteract conflicting KPIs between departments ensuring visibility to the objective of each team. What is important is to establish only a few **key** performance indicators and supporting metrics that a business unit can actually influence and that makes sense to measure. The metrics supporting your KPIs will differ from whether you are chasing app installs, retain users, or turn users into (paying) customers. Each app and each business model is different.

If you want to know more about mobile metrics that app developers use to measure success, there are many resources online such as Braze's article "10 Essential Mobile App KPIs And Engagement Metrics And How To Use Them"[4].

---

4   braze.com/blog/essential-mobile-app-metrics-formulas

Deciding What to Track and Measure

What would you like to measure to understand how the app is being used? Certainly, this depends on whether you are concerned with product management, marketing or the development/technical unit of an app business: Mobile analytics guide marketing teams to create better campaigns. For example; by segmenting users and automating campaigns. At the same time analytics help product teams to A/B test new features and develop the product further. And developers use mobile analytics to track performance and to become aware of technical problems.

Let us take a look at three categories of analytics:

— App Performance Analytics
— Product Analytics & User Feedback
— Marketing Analytics

Again, as we obtain more data, interpreting the details becomes more complicated. Hence, it is important to focus on a handful of KPIs that are relevant to your business goals.

App-Performance Analytics

App performance analytics provide answers to questions such as:

— Why is my app crashing?
— When and on which devices?
— Where do users encounter problems?
— Are any third party SDKs slowing the app down?

Technically, app performance tracking is part of product analytics. It is about understanding how often and why the app crashes and identifying remedies for points of failure. According to data from Apteligent, 30% of consumers indicate that they will consider leaving a brand due to poor app experience[5]. Monitoring your app's performance-based metrics is the backbone of successful optimization. Your task as a developer is to fight any performance issues and optimize the code so your users can enjoy a fault-free, seamless mobile application.

The following KPIs can be used to tweak your app accordingly:

— User device attribution (iOS vs Android)
— Crashes
— Launch time and resume time
— Load time
— Battery consumption
— Error rate
— Number and length of sessions
— UI average response time
— Screen dimensions

Google Firebase[6] is sufficient if you are looking for a simple app performance analytics solution. For more in-depth analyses that can detect bugs or slow data connection, commercial analytics tools are necessary. Public transport apps, for instance, are interested in analyzing the connection speed data over the day. With commercial tools such as *Apteligent.com* or *NewRelic.de* peak times and rush hours can be evaluated.

---

[5] "Retaining Mobile App Customers 3 Strategies to Fight Churn", youtube.com/watch?v=66S9k7plzbU

[6] firebase.google.com

Product Analytics

Product analytics provide answers to questions such as:

— How do users interact and engage with the app?
— At which point and screen does a user leave the app?
— Where do users get stuck?
— What features get the most use?
— How long does a user stay in the app and how many users return to your app?

To craft a data-based product strategy, product teams use analytics insights to build and defend their product roadmaps. User behavior is a tough nut to crack. Knowing how users interact with your product, how certain features are used, or whether they are used at all is essential to offer users a positive UX.

As a product manager you want to look at some of the following KPIs to tweak your app:

— Ratings and reviews in App Store and Google Play Store
— Stickiness
— Daily/weekly/monthly active users (DAU/WAU/MAU)
— Crashes
— Load time
— Number of uninstalls
— Retention
— Session length
— Number of opens

Tracking these KPIs involves using dedicated tools with the capabilities to map user behavior within your app with heatmaps, user flow and screen recordings. We will provide more information on how to track responsibly at the end of

this chapter. The data collected from product analytics can then be shared with development teams, marketing and growth teams to create better UX.

Marketing Analytics

Marketing analytics provide answers to questions such as:

— Which acquisition channel generates the most (valuable) users?
— Which parts of the app lead to conversions?
— How much did you spend on acquiring a user?
— Is there any danger of fraudulent traffic from my traffic sources?

There is an overlap between product managers and marketing managers both relying heavily on user behavior data. The data will be different for marketers, as they are more concerned with measuring the quality of their marketing activities by evaluating the quality of users from respective acquisition sources against each other. Hence, an attribution software provider is at the heart of any marketing setup to review performance and budget allocations for acquisition measures.

As a marketing manager you want to look at some of the following KPIs:

— Installs
— Retention (this could be weekly, monthly or after day 1, day 7 and day 30)
— Store Pageviews
— DAU/WAU/MAU
— Stickiness
— Activation rate
— Conversion rates from Google Play and Apple App Store

- Churn rate
- Returning vs new users
- Total amount spend on advertising
- Push notification click conversion

After clicking on an ad, users are directed to the Apple App Store or Google Play Store. In the stores, it is not directly possible to capture the information as to whether the user is installing the app or not. In order to fill this gap many mobile measurement partners (MMPs) have entered the market around the traffic attribution of app installs.

Why do you need a commercial attribution tracking tool? Google Analytics is not suited at all for mobile attribution tracking because it does not collect traffic from important app traffic sources such as Facebook, Apple Search Ads and ad networks. An attribution tool allows you to attribute any incoming app install to its corresponding source. Secondly, it will also measure your post-install user behavior to give you valuable insights about what your users do in your app, related to the source of the install. This post-install behavior will be measured via events that the user triggers. This information is collected centrally and should be impartially assessed by traffic providers. The market is dominated by MMPs such as *Appsflyer.com* and *Adjust.com*. These tools vary widely in features and functionalities (fraud detection, deep linking) and provide a different pricing model. Read the chapter about User Retention and mCRM in this guide to learn more.

Qualitative Data Sources - User Feedback

Hard, quantitative data is a great source for improvements in your app. Yet, another way of receiving insights into user behavior is combining analytics together with qualitative data via user feedback.

How can app developers actually understand the motives of their app's users? Well, the answer to this lies in communicating with users, receiving feedback and leveraging it to improve the user experience. There are several reasons why you start integrating user insights in your app analytics:

— Validating assumptions you are making based on the information you are receiving from your analytics sources.
— Reducing costs by allocating resources to features that will add value
— Boosting the connection with your users and opening possibilities for more engagement.
— Improving the product road map and UX to prioritize features that matter most to users.

How do app developers actually get user feedback? You may want to consider collecting feedback via:

Apple App Store and Google Play Store

The app store rating is the number of stars you see in both the Google Play as well as Apple's App Store. It is basically showing the average customer rating that your app is getting. iOS and Android developers can program a prompt for users to rate the app and leave reviews. Optimizing the review flow is incredibly important in order to maximize your chances of

getting reviews from your users[7]. You can read more on this in the App Store Optimization Chapter in this booklet.

In-App Feedback

Specific in-app user feedback tools such as *Apptentive.com* can help developers to strategically obtain user feedback requesting it right there on the spot in the app with certain mechanisms such as feature request management or in-app surveys. Create feedback functions, such as contact forms and basic surveys within the app, in easily discoverable locations. It should be convenient for customers to reach out inside the app. Your app is definitely *the* best place to collect feedback. If a user just completed the next level in a gaming app, *now* is the time to ask them about their experience without leaving the app. Nevertheless, do not forget that multiple feedback channels can work together.

Social Media, Forums and Communities

Facebook, Twitter, Instagram and Co. are meaningful channels through which you can collect valuable feedback from your target groups. Once your app has got a considerable amount of traction. If you do not have that traction yet you might want to begin with social listening monitoring mentions of your app (and your competitors).

---

7   "5 Clever Ways to Increase Mobile App Reviews", neilpatel.com/blog/increase-mobile-app-reviews

Choosing an Analytics Data Source

In order for your mobile app to succeed, there are numerous metrics you need to be measuring and analyzing. This means you will need to find a source to actually get your data from. After setting up KPIs and supporting metrics you need to define your sources:

— **Developer consoles**: With App Store Connect[8] or the Google Play Console[9] developers can manage and submit apps, yet they can also acquire information on analytics data.

— **Commercial mobile analytics platforms**: Google's own analytics tool, Google Firebase[10] is the most popular free tool. App developers often choose Firebase because it seems to be meeting many marketing requirements. The actual usability for marketing managers is unfortunately very limited. Mobile app analytics tools saturating the market are adapting to the need and giving app creators innovative new ways to track and understand user behavior. An analysis of your own requirements for such an app analytics tool and comparing the market is worthwhile in any case.

— **Opensource**: there are several opensource mobile analytics offerings that include source code for both the client and the server. Examples include *Count.ly* and *Matomo.org* (previously Piwik).

— **In-house**: it is possible to create in-house mobile analytics software. Various companies have done so.

When it comes to choosing the right mobile analytics tool for your app, you can easily become quite overwhelmed. Some main commercial tool options available today from a marketing and products point of view include the following:

Attribution/Marketing Analytics:

— *Adjust.com*
— *AppsFlyer.com*
— *branch.io*
— *Kochava.com*

Product Analytics:

— *firebase.google.com*
— *Amplitude.com*
— *Mixpanel.com*
— *Count.ly*

Mobile Analytics Technology

Mobile analytics track user behavior and interaction within an app. In terms of technology, we have to distinguish between mobile web that primarily relies on JavaScript and cookies; and mobile apps that require a software development kit (SDK). SDKs contain a package of pre-written code that developers embed in the mobile app in order to measure metrics that are important to your analytics strategy. Different platforms need different SDKs, which exist for Android and iOS.

Mobile analytics SDKs generate a unique identifier for each device. They do not depend on web cookies. This is how analytics tools can track and compile reporting. What does mobile analytics software usually track?

- — Visits
- — Sessions
- — Visitors
- — Pageviews
- — Source data
- — Device information
- — Login/logout
- — Custom event data

Tracking & Handling Data Responsibly

You as a developer are responsible to limit and safeguard the user data that you collect. App publishers have to tell their users when, why and how they are tracking[11] them to avoid legal actions. When a user performs an action in the app the tracking code sends data directly to the data collection server of a respective tool provider. In terms of terminology and the wake of GDPR, "tracking users" have stirred up negative conversation when it comes to user privacy[12]. Respecting your users' privacy means to collect only data from the get-go that is directly relevant to your app's core functionality. Be thoughtful and carefully consider the costs and benefits of collecting data. Grabbing any information beyond that might just lead to bad press and lack of trust with users.

Furthermore, hold on to data just as long as you need to. Just because you need location information, it does not mean you actually need to keep that data forever. Delete it once it has fulfilled the purpose for which it was collected. This helps the potential harm of data breach and other privacy issues.

---

11 Tracking here means recording user behavior within the application

12 "Many popular iPhone apps secretly record your screen without asking", tcrn.ch/2td6gt7

In that regard, it is incredibly important for app companies to sign a data processing agreement/NDA with any mobile analytics software provider.

The purpose behind the tracking and the kind of data collected should be well-informed in the terms and conditions or data policy of your app. Marketing analytics tools such as Adjust have already implemented the "right to be forgotten" with a dedicated opt-out option[13] for users.

What to Consider?

If you want to avoid common mistakes, there are a wide range of topics to consider when implementing and integrating mobile analytics in mobile apps. The following are some of the most critical:

Having enough users/traffic: You need enough data to make qualified decisions. If your app just launched and you have not tested your channels yet, you need to nail user acquisition first before starting with mobile app retention and analytics to feed your retention campaign.

Asking the right questions: You have to ask the right questions from the start and focus on key metrics that are of importance to your department (marketing, product management, development, management). You need to know how to link business results with the factors that drive those results, know what metrics to track and finally, know what value you want to get out of them.

Ignoring vanity metrics: These are metrics that are irrelevant and do not act as an indicator of the success of your mobile app business (e.g. app downloads, social media follow-

---

13 adjust.com/forget-device/

ers). They sound impressive but they do not mean anything and do not focus on user behavior.

Having a tracking concept in place: Implementing a tracking concept is of utmost importance. Most analytics tools are event-driven relying on events to collect data. An event is simply an action by a user such as launching the app, signing up for your product, activating a subscription or accepting push notifications. The tracking concept should not be too complex but functional to the marketing targets.

Customization: Custom event tags augment predefined events, and many mobile analytics solutions provide ways for your app to generate them. You may need to format the custom event messages. If so, pay attention to an encoding of the elements and separators. For instance, they may need to be URL encoded[14] when they are sent as REST messages.

Costs: There are various costs such as implementation, financial, operational and costs related to data privacy. These costs need to be considered and justified and only implemented where the value significantly exceeds the costs, and where the actual and potential costs can be mitigated.

Testing and calibrated results: Avoid trusting and interpreting your data blindly. If you find significant patterns in your data, consult with product managers who understand user behaviors and the product itself. A good practice is to test the analytics implementation at the outset, starting with no users, then one, before testing with more users. Look at latency, accuracy, and reliability of the recorded data.

Betraying users' trust: We have covered this issue under "Tracking Responsibly". In terms of tracking users with SDKs, make sure that you do not blindly "hand over the jewels", but have sufficient rights and access to the data that is collected

---

[14] en.wikipedia.org/wiki/Percent-encoding

Mobile Analytics & User Feedback

by the analytics software provider. Remember to explain to the end-users that the app is designed to record and share information about how the app is being used, ideally in your terms and conditions.

Time, latency, time zones, cut-offs: There is a lag from when an app sends an analytics event to when the information is processed and made available to you. The lag, or latency, varies from near 'real-time' to many hours. You and your business sponsors need to decide how long you can afford to lag real-time events. Also, remember to address globalization issues such as the timestamp of each element. Does the app detect the time of an event according to the device's location, the device's settings or does it use a global timelike UTC time? Some providers, including Google, have cut-offs for when data arrives in order for the data to be reported on.

Visualizing data correctly: Data visualization is about taking raw app data and transforming it into graphs, charts and images. Creating value out of data visualization can be a complex process. That is why good data visualization relies on good design and should be easy to understand for your different stakeholders looking at the dashboard.
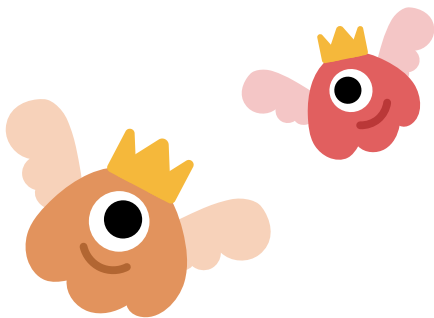
Learn More

This chapter covers the basics of mobile app analytics, which is a field that is ever evolving. It is a complex topic and it is impossible to cover everything. We hope this chapter has whetted your appetite to learn more about mobile analytics. Here are some places to start your ongoing research:

— Amplitude is a mobile analytics tool. They provide a comprehensive guide to mobile analytics: *www.amplitude.com/mobile-analytics*
— Data Stories, a podcast on data visualization with Enrico Bertini and Moritz Stefaner: *www.datastori.es*
— Customlytics' blog provides a series on data warehousing: *www.customlytics.com/en/blog*
— The Digital Analytics Power Hour podcast: *www.analyticshour.libsyn.com*
— The Mobile Analytics Playbook by Julian Harty and Antoine Aymer includes material on using mobile analytics to help improve testing of mobile apps: *www.themobileanalyticsplaybook.com*
— Mixpanel is another analytics software provider and is offering an ultimate guide to mobile analytics: *www.mixpanel.com/topics/what-is-mobile-analytics*
— Big data blog by Reddit: *www.reddit.com/r/bigdata*
— Venture Beat Big Data blog: *www.venturebeat.com/category/big-data*
— A podcast with Christian Eckhardt on how to measure metrics: *webmasterradio.fm/episode/how-to-measure-the-metrics-that-matter-most-and-get-started-with-data-warehousing*
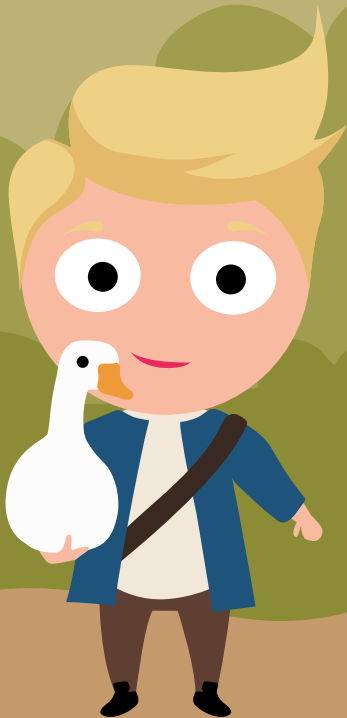
Epilogue

Thanks for reading this 18th edition of our Mobile Developer's Guide. We hope you have enjoyed reading it and that we helped you to clarify your options. Perhaps you are now ready to get involved in developing a mobile app or have discovered new options in the app business. We hope so. Please also get involved in the community and share your experiences and ideas with us and with others.

We said it before, and we cannot say it often enough: If you like to contribute to this guide as a writer or support upcoming editions as a printing sponsor please write to *mobiledevguide@open-xchange.com*. If you are using Twitter, follow the project on *twitter.com/mobiledevguide*.

You can of course also get this guide as an ebook - just check *amazon.com*. Or you simply download the pdf version on our website: *www.mobiledevelopersguide.com*.

About the Book

This project was initiated by Enough Software in 2009 with the aim to spread knowledge about mobile technologies and to encourage people to enter our community or deepen their existing knowledge. In 2018, Open-Xchange (OX) took over the project, Customlytics joined with this edition. Until today, we have given away almost 100,000 hardcopies at events worldwide. Universities and schools in Germany, Netherlands, UK, Spain and South Africa use the book as teaching material. The electronic versions (ebook and pdf) have been downloaded hundreds of thousands of times and the content has been translated into several languages. The book is a non-profit project: the writers, editors, translators and designers contribute their work free of charge.

The Publishers

Open-Xchange

Open-Xchange is a developer of open source communication, collaboration, office productivity, and DNS-based security software and services. With 270+ employees and a global presence Open-Xchange is well positioned to meet the needs of our large Internet Service Provider, Hosting, Telecommunication and Cable Provider audience. Customers include 1&1, Orange, Italiaonline, Comcast, GoDaddy, and Softbank, with more than 200 million users already using our products. The core of the product portfolio is OX App Suite with applications for email, contacts, and appointments. OX App Suite can be expanded to include additional apps such as: OX Drive for managing and synchronizing files, OX Documents for text editing, spreadsheets and presentations and OX Guard for email

and file encryption. Dovecot, the world's leading open source IMAP server, and PowerDNS, a provider of secure DNS services, complete the Open-Xchange product portfolio. Open-Xchange is headquartered in Cologne, Germany, with offices across Europe, in the USA and Japan.

Our culture is founded on a commitment to openness, a rebellious business spirit and the desire to leave the world better than we found it. Effective teamwork is central to our growth.

Learn more about the benefits of working with OX:

🌐 *www.open-xchange.com*
🐦 *@openxchange*

Customlytics

Customlytics is a Berlin based app marketing consulting agency. We enable apps to grow successfully by establishing solid marketing analytics & technology infrastructure. To do so we offer hands-on support as well as strategic guidance in the fields of mobile marketing, product management, analytics & technology integration.

Marketing & Technology: We speak both and act as a mediator between marketing, product and development teams. This enables our clients to successfully manage their own app marketing efforts in the long run.

🌐 *www.customlytics.com*
🌐 *customlytics.com/en/blog*

The Authors & Supporters

This project would not have been possible with the ongoing support from the mobile community. These are the folks that have been involved as authors this time. Some of them are on board for 10 years already, others just recently joined. All are awesome.

Andrej Balaz / Independent Service & UX Designer, Partner at Revealed

Andrej works at the intersection of business, strategy and service design. He combines tools and perspectives from design and market research, bridging both disciplines to discover and define new markets. Formerly at IXDS, Andrej helped clients across various industries such as Bosch, Deutsche Bahn, Fraunhofer, Applause and others to transition from insight generation to building and testing new services.

Andrej is currently a partner at Revealed, a consultancy that empowers businesses to create sustainable growth through an understanding of why consumers really buy and use products. With his team that combines the skills of market and design researchers, product managers and a former criminal investigator, he provided insight into customer goals and behavior for teams at Pipedrive, Asana, Twitter to name a few.

🐦 @Designamyte
🌐 www.balaz.de

Daniel Böhrs / Open-Xchange

Daniel started developing software in 2006 with PHP and web development. During his computer science studies he focused on Java and especially Android development. After obtaining his master's degree he joined Enough Software in 2015, which later got acquired by Open Xchange. In the company, he is mainly an Android developer, although he sometimes helps out in the server development area.

🐦 *@Boehrsi*
🌐 *boehrsi.de*

Dean Churchill / AT&T

Dean works on secure design, development and testing of applications at AT&T. Over the past several years he has focused on driving security requirements in mobile applications, for consumer applications as well as internal AT&T mobile applications. He has been busy supporting AT&T's emerging Mobile Health and Digital Life product lines. He lives in the Seattle area and enjoys downhill skiing and fly fishing.

Oscar Clark / Unity Technologies

Oscar Clark is an author, consultant and evangelist for Everyplay from Unity Technologies. He has been a pioneer in online, mobile and console social games services since 1998. He provided 'vision' for one of the first Online games communities (Wireplay - British Telecom); was the global lead for games at Hutchison Whampoa (3UK) which included (perhaps) the first mobile in-App purchase, and was the Home Architect for PlayStation®Home.

He is a regular columnist on PocketGamer.Biz and Develop-Online, an outspoken speaker at countless games conferences, a mentor for accelerator GameFounders and has guest lectured for several universities. His first book, "Games As A Service - How Free To Play Design Can Make Better Games" is available online.

🐦 *@athanateus*
🌐 *www.gamesasaservice.net*

Sarah Duwe/ Open-Xchange

Sarah is an apprentice for digital media design at Open-Xchange. As part of the mobile design team at OX, she is involved with UI, UX and graphic design for mobile applications. She is also responsible for the design of this guide. In her time off she likes to play video games and to create digital illustrations.

Christian Eckhardt / Customlytics

Christian is the CEO & Co-Founder of Customlytics where he helps app companies to establish a solid marketing analytics and technology infrastructure. Among many other things of course, such as growing the company and making sure his employees and colleagues are happy. Previously to founding his own company, Christian gained extensive mobile marketing and product management experience at Delivery Hero, growing the international app business. Christian's expertise in mobile marketing is pretty varied with a keen interest and focus on mobile CRM, product management and performance marketing. You can also find him frequently on stage as a speaker for various industry events such as App Promotion Summit or Mobile Growth Summit.

🌐 *www.customlytics.com*
in *www.linkedin.com/in/christian-eckhardt*

Julian Harty / Commercetest

Julian's mission is to help people live better lives through mobile technologies. He has worked globally in leadership and engineering roles for Google, eBay, Salesforce, Klarna, Badoo, and many others to help improve engineering practices and deliver better quality software while also improving the fulfillment and motivation for the people involved. He has been actively involved in the modern mobile ecosystem since 2006 and this guide from 2010 onwards. Currently, he is working independently, writing mobile apps & suitable test automation tools, and helping others to improve their mobile apps. He is also trying to complete his PhD on improving development and testing of mobile apps.

🐦 @julianharty
🌐 github.com/julianharty

Linda Harnisch / Customlytics

Linda works as a communications and content manager writing articles and blogs on mobile app topics. She has developed a deep appreciation for mobile technologies making her (and people's) lives easier. As a previous sociology student, she advocates for using empirical data and analytics to shape e.g. marketing and communications decisions. Linda has worked in PR and marketing with a focus on connecting business communities through networking experiences. At

Customlytics she is drawing on that experience by shaping the company's partnership strategy.

🌐 *www.customlytics.com*
🔗 *www.linkedin.com/in/lindaharnisch*

Swen Hutop / swenden

Swen is Managing Director and Co-Founder of swenden. With his digital product expertise, he supports companies in the space of app development. This incorporates conceptual work, programming and a huge portion of consulting. He always strives for clean and easy solutions.

🐦 *@swenden_de*
🌐 *www.swenden.de*

Dennis Kluge / swenden

Dennis is the Managing Director and Co-Founder of swenden.

He consults clients in the field of digital product development. Topics cover conception, implementation and product validation. Everything in a lean way. From time to time you will find him giving workshops and speaking about his experiences and knowledge.

🐦 *@swenden_de*
🌐 *www.swenden.de*

Cornelius Haverland / Open-Xchange

Cornelius specializes in graphic, UI, UX and visual design for mobile applications and other interactive technologies. He was in charge of the layout and design of the previous guides. When not involved with something mobile, he loves to experiment with digital art and illustration.

Ruadhán O'Donoghue / Western Technological

Ruadhán is a web developer based in Ireland. He has worked in web development since 1999, and developed his first mobile web application, a WAP dictionary, back in 2000 when the mobile web was built on WAP and WML, and was browsed on tiny monochrome phone screens. Since then, he has worked in many different roles including Head of Engineering at Afilias/dotMobi. As an editor and contributor to mobile technology site mobiForge.com for over 10 years, he writes articles on mobile web development regularly. He published his first book, "AMP: Building Accelerated Mobile Pages" in 2017. He now runs his own web consultancy, Western Technological.

🐦 *@rodono*
🌐 *westerntechnological.com*
🌐 *ruadhan.com*

Cathy Rundle / RNIB

Cathy is a User Experience Manager at RNIB and together with her team works with companies on their products, websites and apps to improve the usability and accessibility for blind and partially sighted people. This includes design, heuristic assessments and user testing with blind and partially sighted people.

🌐 *www.rnib.org.uk*
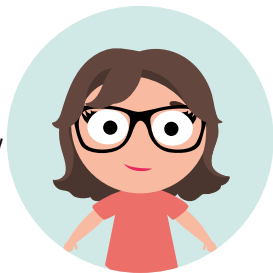
Laura Spikermann / Customlytics

As an app marketing consultant, Laura is making sure that app companies are set for success now and in the future. One of her major fields of expertise is advising apps on their app store optimization activities. If you want to know how to drive conversions and gain visibility in the Google Play and Apple App Store, she is the one you should turn to. Together with the graphic design department at Customlytics, she is developing engaging creative assets for app product pages in the stores - and she has eagle eyes for any askew alignement. Other areas of her expertise include mobile CRM topics and tracking.

🌐 *www.customlytics.com*
in *www.linkedin.com/in/laura-spikermann-205622130*

Marco Tabor / Open-Xchange

Marco started working in the mobile software business in 2005. Three years later, he joined the dream team at Enough Software where he has been responsible for PR, sales, project management and much more. Since the Enough Software team has become a part of Open Xchange in 2016, Marco fully concentrates on his role as a product owner, mainly for mobile apps. He is also the main coordinator of this book project since its first edition.

🐦 *@enoughmarco*

Marc van 't Veer / Polteq

Marc is a mobile app test expert at Polteq and has worked in different test roles for over 14 years. He is helping companies with test process improvement (TI4 Mobile approach), mobile device analysis, mobile test strategy, implementing mobile test automation and giving the in-house developed mobile trainings (mobile app testing foundation, mobile automation, API manual and automated testing). Currently Marc performs research for Polteq on testing AI, is program chair for the Polteq conference and is running projects for multiple companies.

🐦 Twitter: *@marc_vantveer*
🌐 *marcvantveer.niobe.nl*
🌐 *www.polteq.com*

Robert Virkus / Open-Xchange

Robert has worked in the mobile industry for more than 20 years. Having been involved in a number of open source projects, and initiating one of the most successful pre-smartphone mobile app development frameworks, he tries to push the envelope based on open & permission-less standards. In his spare time, he likes to draw, play with his retro computers and spend quality time with his family. Robert has given talks at various conferences such as Droidcon, CeBIT, MobileTech Conference and more. He is currently product manager for e-mail and chat apps at Open-Xchange.

🐦 *@robert_virkus*
🌐 Mastodon: *mastodon.cloud/@enough*

Mladenka Vrdoljak / Open-Xchange

Mladenka is working as a UI/UX designer at Open-Xchange. That means she is engaged with User Interface, User Experience and graphic design for mobile applications, as well as coding. She has been responsible for the layout, design and text editing of previous editions of this guide. For this edition, she was in charge of the text editing. In her spare time, she likes to travel, draw, play the guitar and video games.

🐦 *@_mladenka*

Get your mobile growth infrastructure on track with our consulting services

Strategy & Training

Strategy Development

Mobile App Audit

Workshops

Due Diligence Audit

Marketing Infrastructure

Attribution Tracking

CRM

Deeplinking

BI/Data Warehousing

Marketing Services

App Store Optimization

Performance Marketing

Ad Design

Retention Campaign Managemnet

Product Management

Product Analysis

Product A/B Testing

UX & UI

BI/Data Warehousing

CUSTOMLYTICS

customlytics.com
info@customlytics.com

Please follow us on
Twitter *@MobileDevGuide*.
Thank you!

An initiative by:

OX

CUSTOMLYTICS

A non-commercial, community-driven, independent overview on mobile technologies for developers and decision makers

"A must-read for everyone considering creating and marketing apps."
- Steve on amazon.com

„This guide is awesome. Chapters are perfectly suited for College-level computer science student with just enough technical details to understand the scope of the subject. Also, the edits and additions to the content from year to year make it worth reading every new edition."
- James Hoffman, Teacher at Collège Shawinigan Quebec

"A spectacular piece of work! You will be astonished by how incredibly fast you can establish your presence in the mobile market with the simple steps explained in this guide."
- Daniel Hudson on webtechman.com