The importance of tracking mobile applications

Stef Van Gils

Monitoring of mobile applications is a relatively new concept, there is not much research done to explore this kind of monitoring. However, the data collected from the monitoring can be used to improve the application.
In this article our vision on monitoring mobile applications is presented. The article explains how the design of a monitoring library could look. This design is based upon the Metrics library ([12]). Our design gets compared to libraries that exist at the moment and an overview of the differences is made.

I. Introduction

WITH approximately 1 out of 5 people using a smartphone ([4]), the mobile phone has experienced a revolution. Smartphone applications are used daily and a key property of them is that they have to be responsive almost all the time. This is not easy to accomplish for developers, some delays are only visible when the application is in production and not in the testing phase. At this moment there is no straightforward way to discover this problem. A possible solution is presented in this paper with the help of a monitoring library.

The article starts with giving a global overview of mobile applications and smartphone usage. With this view it is possible to talk about the importance of the monitoring mobile applications, which is best to be kept in mind in the following sections.

The key part of this article is which parts of the application should be tracked. To make the image a bit clearer, this is split into two sections: the monitoring of performance and the monitoring of the usage of the application.

At the end the monitoring library gets evaluated with the lessons that are learned from the previous sections.

II. The impact of mobile applications

In 2015, the number of smartphone users has grown to 1.859 billion users. ([4]). That means that approximately 1 out of 5 people uses a smartphone. Every smartphone consists of a number of mobile applications (referenced to as apps further). The Apple App Store consists of approximately 1.5 million of them, which means that there are a lot of apps already in the open ([5]). This number also means that there exists more than one app for one purpose.

The impact of the smartphone can be seen as enormous. Adults spent more time on their smartphone then on any other digital device ([6]). The smartphone now accounts for one third of the internet usage ([7]). This is very important to keep in mind when reading the rest of the article.

All these properties lead to a couple of questions:

- As developer, how do you distinguish yourself from other similar apps?
- What is the impact of app perfomance?
- Does a user use my app as intended?

The answer to all these questions can be found in the next sections.

III. The importance of monitoring

As developer it is desirable that the software that you write is not slow and works as it should work. To be sure that this is the case, a developer writes tests and concludes that his code works or that he should alter his code. However there are many cases that testing alone is not sufficient. One case is that if data gets sent or received from a server and the server is overloaded, the app appears slow, but the malefactor is the server. This is one example showing that sometimes it is not possible to foresee the bottlenecks of an application. To be able to find out where the issues lie, it is necessary to monitor the application. If the application is monitored correctly, the developer should be able to identify the bottlenecks in the application and solve them in a following version.

Not only is it important to monitor the performance of an application. In many cases a developer or owner of an application wants to know which parts of the application gets used mostly by the users so they have an idea where to improve or alter the application.

A. *Performance*

As discussed before, performance is a key issue in developing mobile applications. Three out of the top ten of most mentioned complaints for mobile applications are performance complaints, namely: *Resource-heavy*, *Slow or lagging*, *Frequent crashing* ([8]). These complaints are the problem of the developer and should be solved to improve the application. These problems could be discovered by monitoring the application. Without monitoring the application for performance issues, the developer has to read the reviews by users to identify the problems, which is in most cases almost impossible to do because of the lack of info that is in the review. If the application is monitored as complete as possible, the developer is able to identify the correct spot in the code where the performance issue lies and solve them in the following version.

B. *Usage monitoring*

There are two reasons for monitoring the usage of a mobile application, namely: to discover which parts of the application gets used and in combination with performance monitoring to find bottlenecks in the application.

The group of persons that wants to discover which parts of the applications the users use are the owners of the application. The main reason they want to know this information is money. If a part of the application isn't used frequently, it is a waste of money to solve bugs in it or invest money to improve this part. The money should be invested in parts that get used frequently to improve the user experience or not even at all. If we look back at the list of most mentioned complaints ([8]), we can see that there is a complaint *Feature removal*. This means that the owners of the application decided the feature gets used rarely. If it was possible to monitor the usage of this feature, the owners wouldn't make that mistake of getting that feature removed. This leads to less disappointment from the users.

Developers could use this kind of information too. The situation gets a lot clearer with an example. Take the situation where the app gets slow when the usage passes n users simultaneously, because the load on the server is too high. If this happens frequently, the developer should increase the server capacity, but if this happens almost never, a solution could be to rent a server when the number of users gets to a critical number. These decisions could save a lot of money.

These situations indicate that there is a need for monitoring mobile applications. A monitoring library could improve the application and the business owning the application in many ways.

IV. What to monitor

Now that there is a clear picture why monitoring is important, it is necessary to check what parts could be monitored. Like the previous section there is a separation of performance monitoring and usage monitoring. This section covers the most important scenarios, off course there are other scenarios possible.

A. Performance Monitoring

The importance of the monitoring of the performance of an application is already clear. In this section it is explained which properties of performance could be monitored to discover problems.

1) CPU Usage

CPU usage on a mobile device is a complicated property. On a mobile device there are a lot of background tasks that are competing with your application for CPU time. It would be wrong to take the percentage of CPU usage, because it works with peaks and differs from device to device. The solution is to use the time an operation needs to complete. The variability on this is less than it on the percentage of CPU usage. It should be somewhat the same over different devices of the same type. CPU usage is a useful parameter to measure, because it measures how much CPU the application uses at a time. If the CPU usage of the device approaches 100%, the device loses its responsiveness and the user notices this, because there would be a significant delay between an action performed by the user and the response of the application. So it is important to keep the CPU usage down to keep the application and the device running smooth.

2) Network Speed

The network speed is different from the internet connection speed. The network speed depends on two different subsystems, namely: the speed of the back end and the speed to retrieve data from the back end. It is important to monitor the network speed, because almost every application uses the internet to send data to a server or retrieve data from the internet. The only thing a developer can't manage is the speed of the internet connection of the device, but the processing speed of the back end is an important thing to manage.

a) Speed of back end: The speed of the back end is the time the back end application needs to process the request. If this process takes a significant time, this delay flows through to the mobile application and gets noticed by the user of the application. It is important to keep this time as low as possible. It is useful to monitor this speed at the back end itself and discover possible bottlenecks or speed issues.

b) Speed to retrieve from back end: There are other issues that can slow the performance of the back end, for example: the server's application server is configured badly, the server can't handle all the requests, etc. The tracking of this happens in the application. Note that this only detects a problem in the server, not where the problem exists. This is how we detect a problem, a system administrator should look further into this at the server side.

The tracking of the network speed consists of tracking the time it needs to retrieve or send the content to the server. It is important to know the type of connection the user has, because there is a speed difference between the different connections (4G, 3G, WiFi, ...). Combining this delay with the delay measured at the back end (discussed in previous paragraph), it is possible to detect a problem at the server side.

3) Battery consumption

If an application uses a lot of battery, the user of the application will notice this. So it is important to monitor the battery consumption, however this is almost impossible. The battery consumption consists of a lot of different, sometimes hard to track, variables: CPU usage, network usage, other applications running, room temperature, etc. It is possible to track the CPU usage and eventually the network usage by measuring how many times there is a network activity in the application and by exploring how long the network interface is used on average with one network activity. It is, on mobile platforms, not possible to discover how many other applications that are running and there impact on the battery of the device and the influence on the application. A battery from a mobile phone loses its capacity faster when the temperature is higher ([13]), so this also has an impact on the battery consumption. However it is currently not possible to measure the room temperature with current smartphones.

B. Usage Monitoring

Besides monitoring the performance of an application, it is possible to monitor how the application is used by the users of the application. It is important for the owners of the application can have a wrong picture of how the users use the app. This could lead to wrong decisions and make the app worse from the users' side.

There are a lot of ways to track the user's behaviour. Here are some examples that are useful to monitor:

- The buttons the user clicks on and how many times
- The screens the user visits and how many times
- The number of search results
- etc.

These are just a couple of examples of what is possible to monitor, but it makes the picture clear.

V. Characteristics of a monitoring library

The challenge is to merge all these requirements into one library. *What is needed to be able to monitor the properties mentioned above?* There is no clear answer for this, every monitoring library has a different architecture. The monitoring library described below is based upon a Java monitoring library, namely metrics ([12]). The library has five monitoring aids, namely: **Counters, Gauges, Timers, Histograms and Meters**. With these, it is possible to monitor most of the requirements of a developer.

A. Counter

A counter is a concept that lets the developer count some property. For example it can be used to log how many times a button is clicked. The counter object should offer the operations to increase and decrease the value of the counter to complete the concept.

B. Gauge

A gauge is a simple object that just consists of a value that is returned by some property. Each gauge can only contain one value to keep it simple. The gauge object can be used for example to log the amount of values returned by a search.

C. Timer

A timer is a concept to track the duration of an event. This can be any event that is available to track. The timer can be used for example to track the duration of a method or how long it takes to retrieve content from the internet, etc. A timer object saves the current time stamp on creation and compares it to the time stamp collected when the timer is stopped. The time between these two is the time the event needed to complete.

D. Histogram

The concept of a histogram is the same as in the world of statistics: to measure the distribution of values. The values are divided over a number of buckets and represented on a bar graph so the distribution of the graph can be seen immediately. The histogram object in the library should only collect one value at each interval to keep it simple.

E. Meter

A meter measures the speed at which an event occurs. An example of a meter is the Unix load calculator ([9]) which calculates the average load in three numbers: the one-, five- and fifteen-minute load average. The meter object can be used to collect such data.

This data should be collected where and when the developer chooses to collect it. It is necessary to collect other data as well, because with this data alone it is impossible to make a correct decision. There is a need to collect metadata as well to support the original data. The types of metadata that should be collected (at least) is: **the name of the application, the version of the application and the device used**. These are the necessary types of metadata that should be collected. There is another type of metadata that is valuable to collect, namely: **the connection type**.

Why are these types of metadata valuable?

A. Application Name

The name of the application is an absolute necessity to collect, because it distinguishes the tracked values from one application to another one (it might be useful to collect more properties like this to ensure no overlap between two applications). If the application name isn't collected, the dashboard doesn't know which values to collect for a particular application and it won't work as intended.

B. Version of the Application

The most valuable type of metadata to collect is the version number. It distinguishes the results from previous versions to results from current versions. The following example will make it clear why it is so important:

Imagine your application has a version A and version B and version B is higher than version A. With the library we discovered that there was a piece of code that slowed the application down in version A, but the developers tried to solve this in version B. After the release of version B you're not completely sure the solution solved the slowness of the code, so it would be helpful to see if version B solved the problem.

So it is important to distinguish the results from version to version to be able to make a correct conclusion.

C. User's Device

The user's device is another valuable type of metadata to collect. This type only has value in tracking performance and not in tracking the usage of the application. *Why?* The usage of a button or visiting a screen doesn't depend on the type of device the user is on, because the design of the application (mostly) isn't different from one device to another. The performance of the app changes in all aspects from device to device. The explanation is that in newer devices the hardware is improved against the older devices (like cpu, network antennas, batteries, ...). This makes it necessary to collect the user's device in the metadata. It has to be relied upon to make a decision regarding the performance of an app.

D. Connection type

Smartphones have (until now) in most cases four types of different connection types, namely: WiFi, 4G, 3G and Edge. WiFi and 4G are sometimes equally fast, but 3G is slower and Edge even slower. The only use case in which the collection of this type of metadata is useful is when a developer wants to track the speed of the network. In that case, if we don't collect the connection type, the results aren't useful, because we merge all the connection types together in one value. When the values of the different connection types are separated, it is possible to get a better view of the speed of the network.

If we combine the data with the metadata, we get an overview of all the properties we wanted to track. This helps the owners of the app to make correct decisions in what to change in the app and what not to change.

VI. Properties of the library

The properties of the previous section can now be combined in one library who collects all the information and sends it to a server to store it. The dashboard application can get the information of the server to generate charts and more technical information. The library should have at least the following properties: easy to use and no significant impact.

The library should have an easy to use interface and should be well documented, otherwise the developer who uses the library wouldn't be able to get the most out of it and the library wouldn't be used as intended.

The library should not have a significant impact on the performance of the application it is tracking. The impact the library could have on the performance is: slowing the app down, clogging the network interface and tremendous battery usage. When these factors have an impact, the library's usefulness decreases, because we want to track these factors with the library, not introduce them. So in that case the library would only be useful in a test environment. It is thus a necessity to evaluate the performance and the impact of the library on an application.

VII. Alternative libraries

Tracking in mobile applications is a relatively new topic. At the moment there are only two mobile application trackers widely available that are worth mentioning: **NewRelic and Google Analytics.**

NewRelic ([10]) is a closed source mobile tracking library with almost the same characteristics as the library described above. The biggest difference is that NewRelic doesn't include feature tracking. The biggest drawback of the NewRelic library is that it is closed source, which means that there is no indication which data gets sent to the server and which isn't what makes this library dangerous when dealing with privacy-sensitive data.

Google Analytics ([11]) is a tracking library that is used across multiple platforms (web, mobile, ...). The difference between Google Analytics and the library described in this article is that Google Analytics only tracks the usage of an app and not the performance of an app.

There is one library worth mentioning other than the two above, namely the **Metrics library** ([12]). Metrics is a tracking library for (non-mobile) Java applications. The reason this library is mentioned is because the core of the library described in the this article is based on the Metrics library.

VIII. Conclusion

Mobile tracking is a relatively new topic, supported by the fact that only two decent libraries exist at the moment. This article showed that, despite the lack of choice in libraries, mobile tracking is an important research field.

The list of complaints we showed earlier shows that there is a need for tracking mobile applications. Without the ability to track the mobile application, there is no way to tell where the app goes wrong without guessing. An app owner should rely on exact data that is measured at the core of the problem to make a decision.

The smartphone market is still growing and will continue to grow. Every week a an average of 4100 apps are introduced in app stores around the world (1.5 million applications in 7 years). This means that it becomes more and more important to distinguish your app from the similar apps that are available or become available. The only thing an app owner can do to keep their users and gather more users is to listen to the complaints of the users. As listed before, many of the most important complaints are possible to track with a decent library and can be solved afterwards.

Mobile tracking is a research field that should be explored further to be able to improve the mobile application field. Mobile applications, developed by third parties, only exist for about 7 years (https://en.wikipedia.org/wiki/App_store), what means that there is room for improvement. Tracking of (non-mobile) software is standard in developing software, so why shouldn't it be standard in developing mobile software.

References

[1] D. Van Landuyt, S. Walraven and W. Joosen, *Variability Middleware for Multi-tenant SaaS Applications: A Research Roadmap for Service Lines*, Proceedings of the 19th International Conference on Software Product

Line, pages 211-215

[2] H. Moens and F. De Turck, *Feature-based application development and management of multi-tenant applications in clouds*, SPLC 14: 18th International Software Product Line Conference, pages 7281, 2014

[3] F. Gey, D. Van Landuyt, S. Walraven, and W. Joosen, *Feature models at run time: Feature middleware for multi-tenant SaaS applications*, Models@run.time 14, pages 2130. CEUR-WS.org, 2014.

[4] Number of smartphone users* worldwide from 2014 to 2019 (in millions). URL http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[5] Number of apps available in leading app stores as of July 2015. URL http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[6] Danyl Bosomworth. Mobile Marketing Statistics compilation. URL http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[7] Alex Hern and agency. Smartphone now most popular way to browse internet Ofcom report. URL http://www.theguardian.com/technology/2015/aug/06/smartphones-most-popular-way-to-browse-internet-ofcom

[8] Kerry MacLaine. Why Your App Sucks: The 20 Most Common Complaints About Mobile Apps. URL http://appealingstudio.com/why-your-app-sucks-the-20-most-common-complaints-about-mobile-apps-2/

[9] Load (computing). URL https://en.wikipedia.org/wiki/Load_(computing)

[10] NewRelic. URL http://newrelic.com

[11] Google Analytics. URL https://developers.google.com/analytics/devguides/collection/ios/v3/

[12] Metrics. URL https://dropwizard.github.io/metrics/3.1.0/

[13] BU-806a: How Heat and Loading affect Battery Life. URL http://batteryuniversity.com/learn/article/how_heat_and_harsh_loading_reduces_battery_life