Developing Real Time Tracking of User Behavior, with Google Analytics for Mobile Phone Devices

Ahmed Hulo & Johnny To

Master's Thesis

Department of Design Sciences
Lund University

EAT 2015

Abstract

Sony Mobile has a quite large internal user group with the purpose of investigating the usability of their mobile devices. The Telephony Domain was using amongst other techniques usability testing to evaluate the usability of their products. The performed conventional usability testing has its limitations e.g. the controlled test environment cannot fully represent real life setting and it's very expensive in terms of time and resources. This ultimately results in less test persons and potentially biased results. These drawbacks cannot be overlooked when it comes to determining the next software release influencing millions of mobile device users.

The purpose of this thesis work was to accommodate this problem by examining real time tracking of user behavior and interaction with mobile devices. By utilizing Google Analytics in the Telephony Domain, we could autonomously gather large quantity of real user behavior data from a natural environment. The investigation resulted in the conclusion that Google Analytics & Google Tag Manager solely weren't sufficient for our purposes in aiding information for user behavior.

To account for this, a tool were developed that were to be called Usage Tracker. Usage Tracker works as a complement to Google Analytics & Google Tag Manager by providing further functionality e.g. finding out how many users are using a feature and how these events are distributed over the users. The resulting statistics from Usage Tracker is also presented in a more intuitive way for easy interpretation. Furthermore, a direct consequence of using Usage Tracker is that less data traffic will be required.

Preface

We want to start off by thanking Sony Mobile, Pär Olsson and Georgios Persson for giving us the opportunity to carry out our thesis work at the Telephony Domain at Sony Mobile in Lund, Sweden. We also want to thank our supervisor from the University Joakim Eriksson, for his advice and feedback throughout the work. Additionally we want to show gratitude to our friend and mentor Mohammad Rahimpur who always lent us support whenever we needed it.

Finally yet importantly we want to thank our families and friends for their support and patience.

Acronyms

GA - Google Analytics.

GTM - Google Tag Manager.

App - Application.

API - Application Programming Interface.

SDK - Software development kit.

GTM container - Queries data to Google tag manager.

Custom Dimension - Custom data sent to GA, which then can be filtered on in the GA web interface.

Frequency - A defined custom dimension for how often a MainFeature will report statistics to Google Analytics.

Sample rate - Percentage of total phone instances to sample from.

OTA - Over the air.

GA web interface - Google Analytics web interface.

Probe - Usage Tracker code used to measure data.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This work was carried out in the Telephony Department at Sony Mobile in Lund, Sweden. The Telephony department is responsible for amongst other things, the Phone application and "Call settings" in Sony Mobile phones. These applications are subjects of constant updating resulting in frequent new additions of features and modifications of features. They have now become very populated and to determine if a feature is still relevant or has become redundant, Sony Mobile uses e.g. usability testing.

The way usability testing is done today in Sony Mobile has its advantages but also its limitations. Our thesis work will be to investigate and develop a way to utilize Google Analytics as a complement to today's usability testing.

1.1 Aims

The main investigation was about how to utilize Google Analytics (GA) & and Google Tag Manager (GTM) to aid in the following questions below, and how to present the data in GA web interface for easy interpretation. The investigation concerned mainly about functionality governed by the Telephony Domain and therefore the resulted product is adapted for Telephony.

After the investigation, we developed the required tool and implemented a selection of probes. The main functionality of the tool is to help answer questions like; is a certain feature used in the intended way, Is a feature even used and why isn't a feature used?

1.2 Method

The work was divided into the following phases:

1. Prestudy
2. Elicitation
3. Investigate which functionalities of GA & GTM to utilize
4. Investigate how to present the data in GA
5. Development

6. Test
7. Evaluation

Since we didn't had much experience of the upcoming work we kicked off with a prestudy phase were we got accustomed to the tools we were going to use and tried to get an overview of the Telephony code. An effective way to broaden our knowledge of GA & GTM is by elicitation. This was done by looking at current applications that utilizes GA and by interviewing people with experience of GA.

After the initial elicitation of GA & GTM, we investigated how to best make use of it for our purposes. E.g. what functionality of GA should be utilized and how to present the resulting data.

In the development phase, the requirements for the tool had been elicitated and this phase was about realizing the product. The testing was done in parallel with the development and at later stages evaluation of the product was also included.

1.3 Limitations

The project were affected by the following limitations:

- Time constraints
- Avoid disturbing the bring up
- Limit on data traffic we're allowed to send

Time Constraints
The thesis work is time limited and when carrying out work for a big company where quality is of uttermost importance and where changes has to undergo several phases before getting approved, it's a necessity to plan with good marginals.

Avoid disturbing the Bring Up
To avoid obstructing the Bring Up we'll be limiting our modifications to packages that are less sensitive to Android upgrades.

Limitation on how much data traffic we're allowed to send
The data sent to GA will be sent over the network. Since more than one application can be utilizing GA simultaneously and GA doesn't take in consideration what type of data connection that is currently being used, a limit on how much data traffic that can be sent each month has been set.

Chapter 2
Theory & Background

2.1 Usability Testing

With an increase of consumer electronics in our everyday life and letting them populate our homes, technology has indeed become a large part of our lives. The advancement in technology is occurring in a faster pace and as consumers, we are expected to be able to follow this fast trend. In addition consumer products are also getting more advanced and the usage of the products tend to follow the same path resulting in loss of many potential customers who aren't able to keep up.

To reach out to all possible customers a well thought user interface with a well thought user interactive experience is crucial for a product especially when competitors' products are just one click away.

The usability of a product has therefore a big impact on our decision when buying a product from a certain brand over another. Let alone mainstream consumer products usually have more than one manufacturer and their end products are often quite similar in functionality. This makes the subjective preference of the customers more important than their objective preference and it's usually the deciding factor for their decision. The most obvious way of implementing this in the development process is by having frequent usability tests throughout the whole development process.

ISO 9241-11 Usability is defined as the "Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [4].

2.2 Benefits of Having Frequent Usability Testing

The main goals with usability testing are to identify any usability problems and to gather quantitative and qualitative performance data to determine how satisfied the test subjects are with the product.

Users are usually not sure about their preferences for a product and are bad at expressing into words what they want for a product. Let alone manufacturers are often underestimating the difficulties in their product by regarding them as common sense.

By having usability tests we can analyze users' behaviors and better understand the difficulties of our products and improve the flaws. The following are consequences from having regular usability testing [5]:

- The products becomes easier to use and more useful to the target audience.
- Minimizes risk and cost by early determining design problems and eliminating them.
- Increase in customer satisfaction will result in a decrease of money spent on customer support.
- Establishes the expectation that the company produces high quality products that are also easy to use.
- An increase in sales and happy customers results in a higher probability of repeat sales.

2.3 Methods of Usability Testing

Usability testing is a term usually used when referring to techniques used to evaluate a product or system were people from the targeted audience are involved. Usability is used to describe how easy a user can use an object to fulfill some tasks. The usability of an object can be tested in many different ways, some of them are described below [8].

2.3.1 Expert Reviews

A person who is an expert in the field of interaction design evaluates the product and provides some feedback. This is very important since this expert's opinions are supported by theories and studies done in the field [5].

2.3.2 Prototyping

A prototype can easily be constructed and tested for its functionality, performance and output. New functionality and ideas can be demonstrated with the prototype and early user feedback of the product can be elicitated. A big advantage with prototyping is that it decreases time, money and risks for the project, since shortcomings in the design can be early detected and these are cheaper to fix the earlier they are discovered [9].

2.3.3 Observations

Users aren't always sure about what they are doing and why they are doing it, therefore if asked there's a big risk that the answer will be wrong. Rather their behavior can be analyzed while performing some given tasks. The session can be recorded and the tasks can be followed up with a briefing to clear out any uncertainties. Observations is an effective way to reveal common flaws but unfortunately not as effective when it comes to unpredicted critical problems since it's unlikely they will occur during an observation [9].

2.4 Internal Usability Testing in Sony Mobile

The internal usability testing performed in Sony Mobile are of various types, these are among others observation, prototyping and expert review.

2.4.1 How it can be performed

One of the types that are performed is observation and works in the way where we have a few test persons in a controlled environment such as a lab. These test persons are then put into different real life scenarios where they are to perform some given tasks. While carrying out the tasks they are encouraged to think out aloud, for easier analysis of their actions.

In this way the perception of the product from the test person's point of view can easily be analyzed. The test session is also recorded for further analysis. After finishing the tests, a debriefing with the test leader together with the test person will be held to discuss the test and the product.

Different user profiles are used depending on if it's a basic feature or a more advanced one. The screening process is to find the best representative users for the product. This selection is of critical stage, if unrepresentative test persons would be used the results would be of bad value no matter how much effort is put into the tests [5][6].

2.4.2 Advantages

Greatest advantage of usability testing of the type observation is that if a test leader have any questions regarding their behavior they can directly inquire the test person about it and receive their subjective opinion. A debriefing can be held after conducting the tests were questions about their actions can be discussed e.g. motive for doing things the way they did and any uncertainties that they felt during the tests. The test person will also have a chance to talk about their user experience in detail and the test leader can have their opinion on alternative prototypes. These are all valuable subjective information to be taken into account in the improvement of the design of the product [9].

2.4.3 Limitations

No matter how much effort we put into these type of usability testing the result will never be good enough to give a sure conclusion. Reasons are listed below [5].

- The artificial environment where the tests are being held is not a perfect simulation of real life usage of the product.
- Test subjects can't represent all of the users.
- Unable to simulate all possible outcomes therefore it's not possible to tell that the product will always work no matter situation.
- In some cases usability tests may not be the most optimal way of testing due to e.g. time, cost and availability.

The greatest disadvantage of utilizing observing as an usability method is that it's very expensive. It requires a lot of resources and time that can result in a small number of test persons that may not be enough to represent the users. To receive feedback as accurate as possible the test person must be in an everyday environment while performing the tests. Real life usage can't be simulated perfectly, instead the test person is given an scenario resembling real life where the product would be used. Even the knowledge of the test person that they're performing a test on has an impact on their performance, due to that the state of concentration of the test persons will usually be much higher than normal.

The test persons are encouraged to try to finish the tasks by themselves but if they are to be stuck and unable to proceed they are allowed to ask the test leader for help or hints. Test persons know about this but often asks for help unconsciously let alone sometimes even test leaders give hints unconsciously because it's of human behavior to help. In the end, the results from these tests may not even reflect on real life usage. This is one of the reasons that makes GA a very suitable candidate as a complement because it doesn't suffer from the same limitations.

2.5 Google Analytics

GA is an API created by Google and it can be integrated in websites or mobile applications. I.e. in the case of using it together with an mobile application it can help with answering some questions about user behavior e.g. is a feature being used, from where is it used and what crashes were thrown when the application crashed [1].

The platform is divided into four components, these are:

- **Collection-** collects user interaction data from e.g. websites or android applications. It can i.e. calculate how many times a button has been pressed in total by all the users.
- **Configuration-** Provides access to the configuration data. To manage accounts, permissions, filters etc.
- **Processing-** The data from the user interaction is processed together with the configuration data.
- **Reporting-** Provides access to all the processed data.

In Figure 2.1 is a diagram made by Google that describes the relationship between the components and the APIs[3].

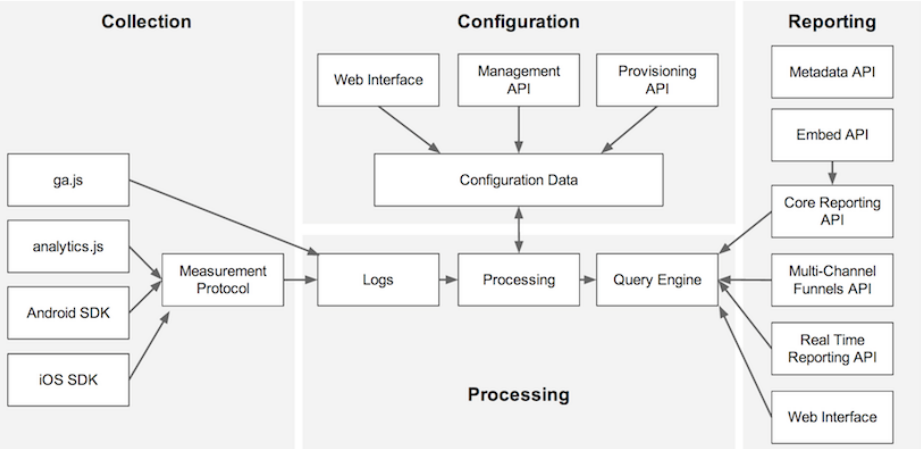


Figure 2.1: Overview of the Google Analytics Platform

Google Analytics is a very capable tool with the capacity to measure heaps of data through user interaction and then generate detailed statistics in e.g. a web interface. For our purpose, which is to track user patterns for improving the user experience, we'll use GA for Android V3, which is a free API provided by Google. GA for android V3 biggest advantage over previous versions is that it can be used together with GTM. GTM for android is a tool capable of modifying variables in the application without the need to access the source code of the application. Values implemented with GTM can easily be modified through a web interface, which enables quick changes over the air. To enable GA and GTM for a certain application the common library for GA & GTM has to be imported to the source code and additions of code has to be made [2].

2.6 Google Tag Manager

A big problem with mobile applications is that it's final after release. If there's some parameters or something that needs to be modified a new version of the application has to be created for people to download or upgrade. I.e. if a mistake would be discovered in the source code, a decimal may be missing or something. A mistake of this caliber would be easily tend to but to release this new version with the bug fix, chances are that it will have to go through all the developing phases again to secure approval from the company before it can be shipped out to the end users.

This is rather a waste of resources and even when the new version reaches the end users, they may not comply with upgrading to this version simply because it doesn't include any new cool features. As a result various different versions of the application may be out in the field and in the end it may be difficult to know which version is actually most commonly used. With the use of GTM a wide variety of parameters can be modified OTA in the GTM website, the new

modified parameters will then be downloaded automatically to the application of the end users.

GTM works in synergy with GA by making it possible to push configuration values to GA via the website. Which means that GA can be remotely controlled OTA and the end users will get the latest version available in less than 12 hours since GTM updates it's container to the latest available twice a day. GTM can also be used as a fail-safe for GA if we want to stop all GA activity [2].

2.7 Usability Testing with Google Analytics

The key benefit with GA is the ability to receive real time feedback from a natural environment in comparison to many types of usability testing where feedback is received from a controlled environment. In a controlled environment peoples concentration rate is usually much higher than normal so the results may not reflect on real life usage [5].

The second thing that makes GA such a great complement to the type observing is that it's very cost efficient and really easy to maintain, allowing for a very wide test group. It also provides a very simple web interface that only requires a login with the right privileges to access.

Deprecated versions of GA suffered from a big problem by not having an effective way to be switched off. For example if a mistake were made resulting in illegal information about the users being gathered, there was no mean to switch it off. Google has since then attended this problem by incorporating GTM to GA. GTM basically solves the problem were a mobile application is frozen after its release.

2.8 Elicitation

The process of finding and formulating requirements is called elicitation. There are various techniques used for elicitation. The following are the ones most effective for our purposes and were all used to cover as much as possible [10].

2.8.1 Interviewing

Interviewing can be used to gather many kinds of information, depending on whom is being interviewed and what the questions are about. It's a good way to acquire knowledge about current work and current problems [9].

2.8.2 Brainstorming

Brainstorming is a technique used to elicitate new ideas in a short amount of time. This technique doesn't involve any participants from the targeted audience and is usually done within a group. The participants will then share all their ideas indiscriminately regarding the topic. No ideas are considered bad in this stage because even the most unexpected idea can turn to something good [11].

2.8.3 Stakeholder Analysis

The stakeholders are the ones that'll be the deciding fact for the product's success. These are i.e. the future users of the product. Hence, it's important to take their interests in consideration.

Goals and key issues for a stakeholder analysis are to find out e.g. who are the stakeholders, what are they expecting of the product, what risks and cost do they see and what solutions do they see. This information can be gathered by either having meeting with all of the stakeholders together or having meetings with each stakeholder separately [10].

We also booked a meeting with the architect at Sony who is in charge of the area that deals with the addition of Google Analytics in applications.

Chapter 3
Prestudy

3.1 Project Plan

Since our University and Sony Mobile hadn't discussed much about what work we would carry out. We wanted to minimize potential misunderstandings by starting out our work with a detailed project plan. This project plan would then act as a contract to get a confirmation from both Sony Mobile and LTH that the work we would carry out is the one intended from both parties.

The project plan consisted of the following information below and a time plan describing what work we would perform and how we are disposing our time.

- **Why? -** What is the main objective of our thesis work?
- **What? -** What is the work that will be performed on the project? What are the major deliverables?
- **Stakeholders? -** Who will be involved and what will be their responsibilities within the project? How will they be organized?
- **When? -** What is the project timeline and when do we have our milestones?

3.2 Presentation of the Project Plan

Proceeding from getting the initial project plan approved by our mentor from the University, we notified our section manager for a presentation to be held for all the parties involved in our thesis work.

The purpose of this meeting was to operate as a stakeholder analysis and to clear out all the potential uncertainties and receive feedback from the project plan that we handed out prior to the presentation.

The main context of the feedback from the meeting was that regular follow-ups were requested and that settings menu for Phone application was of highest priority.

3.3 Elicitation of Google Analytics

In the beginning of our thesis work, we had very little previous experience of GA and Android. Therefore, we chose numerous elicitation methods to broaden our scope of knowledge. The following elicitation methods were chosen interview, brainstorming, stakeholder analysis and looking at similar applications.

We started of the elicitation process with interviews and looking at similar applications because both of them are an effective way to receive large amounts of information in a short amount of time. These two methods are also effective in finding out what previous work involving GA has been done in the company. Starting off with the interviews also gave us the opportunity to further the stakeholder analysis and since we chose to interview mostly people with previous experience of GA we could also learn from peoples past experiences and get inspired by them.

The interviews were of semi-structured type and split into two parts. The first part consisted of prepared questions and the second part was designed for an open discussion. By having this setup, we hoped to receive answers to both specific questions and information that could not cover in the first part because we lacked domain knowledge.

These came to be a great source of inspiration and were our stepping-stone when we later discovered that we had to design our tool to be called Usage Tracker.

3.4 Prototyping

Looking through Sony's wiki page, we found some internal information about GA. Most of it were introductions and guidelines to get people started. In the guideline, a workflow on how to integrate GA for an application could be found. The workflow was pretty straightforward and is described below.

- Add Google Analytics library to the project
- Implement
- Register for GA & GTM accounts
- Evaluate the GTM container and verify that the output in GA looks as expected. Iterate until satisfied
- Go Live!

Since both of us had minimal experience of GA and Android programming, we believed the quickest way to get accustomed was through prototyping by creating a simple Android application while following the reference workflow to include GA & GTM. The application we created was very simple and consisted of a main screen with two buttons.

The first button led to another screen containing a single button used to get back to the main screen. The second button in the main screen displayed a toast with the text "Button 2 has been pressed", when pressed upon. Since this was only for test purposes, the Google Analytics account was signed with our private credentials. For our test application we didn't utilize much functionality of GTM

we basically used it to query data to GA.

Being our first time coding an Android application it took us some time before we got a hang of it, but the output generated from the prototype were more extensive than we anticipated making us realize the potential of using GA.

3.5 Looking at Similar Applications

In the internal wiki page "guidelines for using GA", there was a reference to a library that utilizes GA named GaGtmHelper.jar. This library's purpose was to make it easier to get started with GA and basically consisted of utility classes. Fortunately, it had recently been updated to utilize GAv3 which served our purposes perfectly. The developer was also kind enough to make GaGtmHelperTest.java, which is a reference implementation of GA using GaGtmHelper.jar.

We pulled the project from the GIT repository and ran it on our phone. The application looked similar to our prototype, but had more buttons covering more functionality of GA. Although the GUI of the application looked very similar, the implementation differed a lot. By utilizing GaGtmHelper.jar, the implementation of GA became much more efficient.

3.6 Meetings

After having everything set up and prepared, we booked a few meetings in Sony Mobile with people that had previous experience of GA. The objective of these meetings were to learn from their experiences and from their past mistakes. It came to our knowledge that other teams within Sony Mobile were also currently working with GA, therefore a meeting with one of the developers were scheduled for a briefing.

We were shown some examples of how they had utilized GA and were also presented some of the results. After running GaGtmHelperTest.java and looking at how GA was implemented using GaGtmHelper.jar we concluded that GaGtmHelper.jar would be very useful to us. While studying this library some questions arose so a meeting was scheduled with the developer behind this code. We used this meeting as an opportunity to learn about his thoughts behind the utility library while also trying to get a deeper understanding of overall GA.

3.7 How to Integrate Our Modifications

We want to include GA in an application used by millions of people. For an application with a user group of this magnitude, the quality of the product is of great importance. The Phone application is the main application for any phone and a quality defect could potentially render millions of people unable to make a phone call. An unstable phone application could also potentially stain Sony Mobile's reputation of making quality products.

For this reason, a good deal of quality aspects had to be taken in consideration for the application to work as well as possible. In order to keep it that way we need to implement GA in a way that'll not contaminate the current code. This means that our modifications will have to be approved by the architect of Telephony and then undergo strict tests to verify that our code haven't affected the overall quality. We also need to adhere to rules set by the logging board to guarantee that the legal aspects are met.

Sony have been using GA for some of their decoupled applications. A decoupled application is autonomous and unaware of other applications and the operative system. For a coupled application such as the Phone application, this is not true. The Phone application in Sony Mobile patches the original android vanilla phone application, which makes it a subject to Android changes. This means that a bad implementation of our third party modifications of phone will be more sensitive to Android changes, resulting in that each time a new android version is released unnecessary many modifications have to be made for it to be compatible.

3.7.1 Implement a Service

The code that we had access to and were allowed to make changes in were all in a project called Telephony. The addition of Google Analytics may not contaminate the current code therefore our first scheme was to isolate GA by using a service. The definition of a service is:

" A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background." [13].

This seemed to be a good idea to let GA operate in the background. All required classes for GA were added to the service and some testing were performed to try it out. A meeting was scheduled with the architect of Telephony and our supervisor to discuss this scheme.

It was pointed out that the service was persistent. The Telephony application is as default a persistent application, which means that it is always running. Let alone the fact that using a service the Telephony application wouldn't be offloaded of GA, the conclusion was that we might as well discard the use of a service.

After some discussion we came to the conclusion that it would be wiser to import GA libraries to the Telephony project and simply add a new package in the Telephony project to store all GA classes. Another idea we found positive was to extract the logic from the probes to make them smaller therefore less code to be inserted when adding a probe.

3.7.2 New Design of the Telephony Package

Later in the project, an Android upgrade split the Telephony project into two separate projects. This had a big impact on our work and our previous scheme were no longer feasible. Let alone we had to relocate all the corresponding places to insert the probes. Fortunately the two packages were not independent of each other, there exists methods for the two projects to communicate with each other and a common directory that both project had access to.

There is a class in one of the divided projects (P2) that can be used to send an array of strings to a class in the second project (P1), where the event then will be taken care of depending on type. Since both projects were going to use GA, to prevent importing the same GA libraries to the two projects our new scheme would come to make use of the above functionality.

In the new design a package with all the classes were created in P1 where the GA libraries were imported. A class Constant.java needed access by both projects and were therefore moved to the common directory. P1 contains almost all logic for the probes. Probes will be placed in P2 and when triggered they will be queried to P1 with an array of String. This information are then processed in P1 to identify which probe that were activated and then queried to GA.

Chapter 4
Usage Tracker

4.1 Task List

The scope owner prepared a list of tasks which we prioritized. The tasks with high priority have been listed below. Table 4.1

Table 4.1: Task List

High priority tasks	**Questions to answer**	**What to measure**
After Call Screen	Which options are used and how often?	# of activations % of total calls & reason for ended call
Answering Machine	How many uses it and is it activated manually or automatically? Does people deactivate it?	# and % of launches Manual or automatic launch
Call settings	Which settings are end users changing frequently and from where?	Which settings are changed? How often are they changed? From where are they changed?
In Call Screen Button Panel	Which buttons are used frequently by users	# of button clicks
Reject Call Messages	How frequently are they used and which ones are changed?	# of activations # of changed messages
Bluetooth Headset Audio Routing	How do the users answer, end or reject a call?	# of answered, ended or rejected calls from both headset and phone
Wired Headset Audio Routing	How do the users answer, end or reject the calls?	# of answered, ended or rejected calls from both headset and phone
Route Sound	When the users answers a call with bluetooth coonected, how often do they re-route the sound to the phone earpiece and not the headset?	# of times the sound re-routed to phone earpiece # of times the sound re-routed to headset

4.2 What to Utilize of Google Analytics

GA is a very powerful tool capable of gathering many kinds of statistics e.g. Application views, crashes, events etc. about the application. Those individuals ending up as part of sample needs to be having acceptable level of data traffic sent, even in unusual network states such as roaming.

Therefore, we chose to only collect data that we are to make specific decisions on. These are data relevant for the tasks in the task list received from the scope owner. Event tracking in GA is to be used to collect data about how users interact with the content of an application. An event is a user interaction with the content of the application that can be tracked independently. These can e.g. be clicks, page views, downloads and video plays. For our purposes we want to measure how many times something has been clicked and how many times a particular sub feature has been invoked which fits in the example of when to utilize Event tracking [14].

4.3 Why We Need a Complementary Tool

Since the reference implementation has an example of Event tracking, we originally intended to implement GA by adopting the reference implementation with the use of GaGtmHelper.jar. In our case, this would mean that in terms of usage statistics we would only measure the total number of times something has been activated.

After receiving our tasks from the scope owner we realized that by only using the reference implementation it wouldn't be sufficient to answer the relevant questions specified in the task list. The way GA was implemented for our prototype, the statistics generated by GA would only be presenting us the total number of times a certain sub feature had been activated. This number would be accumulated by all of the users together and we wouldn't by any means be able to calculate out exactly how many times an arbitrarily individual had activated something. No indications if the number of activations are evenly distributed over all the users or if there exist a small percentage of users that made up for a large portion of the total number.

The majority of the tasks were not about finding out how often a feature was activated but instead they wanted us to find out how often something was used and how many uses it. These two questions may sound similar but hold very different meanings and would amount for us to create our own Usage Tracker.

4.3.1 Example

Take for an example were we have button B1 and button B2 and we want to find out which of the two buttons most users are actively using. GA using the reference implementation would only be able to present us the number of times B1 and B2 have been clicked upon from a total number of users' perspective.

Let's say B1 have received 500 clicks and B2 has received 700 clicks by the total number of users. This tells us that B2 have been clicked upon 200 times more than B1, what it doesn't tell us is how many users and how many times an individual user have clicked on B1. Therefore, there's no way to tell if the usage

of the feature is even amongst all of the users or if it's only a small percentage of the total users that represents the accumulated value.

4.4 Development of Usage Tracker

4.4.1 Elicitation of Usage Tracker

For the development of our Usage Tracker, a new phase of elicitation took place. Looking back, we had come across with Usage Tracker when interviewing employees about GA. Back then, we didn't put much thought into it because we didn't realize it would be of much use for us.

Having a better understanding of the limitations of GA we took another look of the before mentioned Usage Tracker hoping that it would fill out the missing gaps. It had similar functionality as what we were looking for but as expected since it was developed for an application very dissimilar to what we were working with, the implementation and the functionality were also very different. However studying this code was very profitable and was a good base for designing our own Usage Tracker, which also happened to be named Usage Tracker.

The strategy behind Usage Tracker is that instead of reporting to GA each and every time something has been activated a counter would instead increment and accumulate over a specific time period i.e. one week, and then only report once after that time is due. We adopted this logic and modified it to fit our purposes.

4.4.2 Brainstorming

In this phase of our thesis work, we had enough knowledge to start a Brainstorming. The Brainstorming was amongst the two of us and main topic was to come up with requirements for our Usage Tracker. The requirements would come to involve everything from design to functionality. When a brainstorming session is followed up with a prioritization of the ideas, the ideas that receives low priority may be forgotten and left out. Therefore, we chose to keep all of our ideas and sleep on them, giving them a chance to grow. The brainstorming was then continued by an expert review with the architect of GA in Sony Mobile.

4.4.3 Context from the investigation

Instead of reporting to GA each time a button is pressed, we need to develop a tool (Usage Tracker) that stores the number of times something has been activated over a certain time period. This specific information should then be reported to GA only when its time period (Frequency) has been reached.

In doing so, it would be possible to look at a certain time interval, e.g. a week in GA and be sure that there is not two events in the data that is reported by the same user. By making users who haven't used a certain sub feature report 0 we will be able to differentiate between people who have and haven't activated (used) something. With this information in hand, we could calculate numerous information such as the percentage of people that are actually using a sub feature.

This solution is not only for measuring clicks but also effective whenever there's a need to measure the frequency of content. I.e. this can be used to measure how many times answering machine has been triggered both automatically and manually. A pleasant consequence of using this strategy for collecting and reporting data is that the amount of data traffic will decrease considerably since it will only report once for each sub feature after a given time period instead of every time it has been activated.

4.4.4 Design of Usage Tracker

Since we are to develop a tool that is intended to be used by people who've not necessarily been involved in our work, we'll be opting for simplicity. For this reason our implementation will follow the guidelines set up by Sony Mobile and follow the reference implementation as much as possible. The code should be as easy to understand, as it is easy to modify for future extensions.

GaGtmHelper.jar has been a common start ground for many GA users within Sony and has functionality that simplifies reporting to GA. Our Usage Tracker will also follow this trend by importing this library and follow the reference implementation for the GTM container opening used to query data to GA. End users will give their consent of sharing usage information by having the "Send Usage Info checkbox" enabled. Depending on the status of this checkbox, all GA activity will either be enabled or disabled. The requirements for the basic functionality for our tool had been established and for the more specific requirements, we split them up into their respective problems for further investigation.

4.4.5 Reporting to Google Analytics

Sub features populate a main feature. To ease readability in GA we've grouped all sub features to its corresponding main feature. Each main feature will be given its own Frequency determining when all of its sub features are ready for report. Reporting all the sub features simultaneously will ensure us that all sub features under a main feature has reported the same number of times.

A timer class will be responsible for the actuation of a check that loops through all the main features. This check will occur when the phone is turned on and after each 24h interval. The check is to determine if a main feature is ready to report by looking at its corresponding Frequency defined in GTM. If the time interval has been reached all of its sub features will have their accumulated values reported.

4.4.6 How the Data Are Stored in the Phone

The accumulated data is saved in the java structure HashMap that is not persistent and we don't want any data to get lost when switching off the phone. Therefore an efficient way of saving the data between reboots needs to be implemented. The original Usage Tracker solved this by saving the data in SharedPreferences so naturally we tried that also for our implementation. SharedPreferences stores private primitive data in Key-Value pairs. Data can be read efficiently if you know the key for the data. Since it only allows for storing primitive values, it's difficult

to store large data, as each value needs a key. Because of the way we're structuring our data and SharedPreferences has no efficient way to structure data and retrieve a specific query of data we considered using a database instead.

Storing the data in a SQLite database the structure of our data could be preserved. The data structure is managed by the database and we can write SQL queries to get specific sub sets from the data. This makes it possible to search and filter out explicit content. Managing and searching after large sub sets of data influences the performance so reading data from a database can be slower than reading directly from a HashMap.

For this reason and to remain our code as low complexity as possible we decided to just serialize the HashMap containing all our saved values and write it to a file. The file will be saved to the internal storage making it private for the application and inaccessible by the user. This keeps the integrity of the file and after uninstalling the application, the file will be removed along with it.

4.4.7 Track Usage of Checkboxes and Settings with Multiple Options

What we came up with in our Brainstorming session wouldn't suffice for tracking usage of Checkboxes and settings were we have multiple options i.e. Equalizer. For features of this type, it's more relevant to measure which setting is enabled rather than how many times it has been invoked by the user. End users don't change settings of this type frequently. When that occurs it's usually because an end user only tried the different settings before settling with the preferred one. The preferred solution to this would be that instead of storing the number of times they have changed the specific setting it makes more sense to read the value of the setting prior to reporting the data to GA.

4.4.8 Grouping of Values

Usage Tracker will report the accumulated number of times an individual have invoked a certain sub feature over a given time period. To make the data more readable in the Event reports in GA web interface we'll create different slots with varying sizes for values to be grouped into before reporting to GA.

The potential slots for a value will be 0, 1, 2-5, 6-10, 11-15, 16-20, 21-40, 41-60 etc. 0 and 1 are presented on their own because it's important to get these two specific values. 0 means that the sub feature haven't been triggered or invoked over the given time period. This gives us the information needed to calculate the percentage of users who actually uses a certain feature. For the lower values, we chose to have intervals of size 5 and for larger values intervals of 20. In the report, we'll be able to tell by the value of the slots how many individuals have fallen into each slot, which describes the number of times a certain sub feature have been invoked by the individual.

4.4.9 What Should Be Remotely Configurable OTA through GTM

We want to keep every main feature independent from each other so we can treat them separately. Therefore, each main feature should have its own sample rate

and sample Frequency that are modifiable OTA. Being able to change sample rate provides the functionality to chose percentage of total phones instances to sample from.

For our purposes which involves an application that is considered to be always active, a small value on the sample rate should suffice. For this we chose to start off with a sample rate of 1%. This parameter can later be modified OTA and will also be used as a kill switch when GA activity is not desired. A kill switch is achieved by setting the sample rate parameter to zero.

Frequency is a custom dimension that we have defined. Each main feature will have its own Frequency determining the time limit. The time limit is the time a main feature will let its sub feature accumulate over before they are reported. This would be useful in e.g. the case when we've collected enough data for a certain main feature and don't want to waste users' data traffic. Another example is if we realize the specified Sample Frequency for a certain main feature is too short or too long, it could easily be modified OTA to desired time interval.

4.4.10 Custom Dimensions to Define

The GUI and functionalities in Sony Mobile phones between models are quite similar to each other but they are not persistent. Many aspects such as phone model, software label, country and operators influence the GUI and the functionalities of a phone device. A feature may have been recently introduced and only the phones with the latest software label may have it and some functionality can vary for different phone models.Countries and operators also have an influence on the functionalities of a mobile phone. These authorities may have different rules and restrictions resulting in changes in the phone content.

Therefore, a feature may not exist in all mobile phone devices even if the model is identical. If the functionality exist, it doesn't guarantee that the feature is accessed or looks in the same way. If phones with a specific functionality and GUI shows odd usage statistics compared to other phones with similar functionality but with less similar GUI, it suggests that the design is at fault. To be able to do this we need to be able to filter out explicit data showing behavior for a consistent system. That's why we need the following Custom Dimensions:

- Frequency
- Sample rate
- Country
- Operator
- Model
- Software version
- Phone model

4.4.11 Implementation of Usage Tracker

In this stage when we had decided all the criteria for the building blocks for our Usage Tracker, we had already fallen behind according to our initial time plan. It was primarily due to our lack of experience when it comes to deliver quality code and another reason for our delay was that we weren't anticipating the need to develop Usage Tracker.

In an attempt to get back to the initial time plan, we got into a rush that resulted into implementing Usage Tracker quite straightforward without any consideration to design principles. Putting it through an initial phase of testing it seemed to do what we expected from an alpha version but after considering the UML generated from the code, we quickly realized that the code structure was a mess. It didn't follow any design principles and we discovered a few circular dependencies. Maintaining our code would be too much of a hassle let alone no one would've the patience to learn the architecture of it. After discussing with our supervisor we concluded that the time plan were just preliminary and not final. This time around, we decided to try with a new approach and regardless of the time pressure, we would do it in the right way.

Instead of wasting time on implementing something that may be of bad design, we would rather follow an iterative model before implementing.

- Analyze the design
- Generate an UML
- Discuss the design

Naturally, the first step would be to analyze the design. Then generate an UML diagram of the code skeleton and use it as a basis to discuss the design together with our supervisor and the architect of Telephony.

The discussion was mainly about motivating our decisions. Since they have far more experience in programming than us, it would be of big help for them to know the idea behind our design. It's possible that the underlying thought for our design was good but we chose a poor way to execute it. From these meetings we could get valuable information in short time and take in consideration their feedback to the next iteration.

It took a few iterations before they were pleased with our design and then it was finally time to implement. Having a well thought design certainly made it easier to implement and resulted in less code and classes compared to our first implementation. The code became much easier to understand and open for extensions. The generated UML- diagram of the final design can be seen in Figure 4.1 below.

After the implementation, we carried out our own verification tests in parallel to the implementation to test the added functionality and to ascertain any unexpected behavior. Many modifications were made to accommodate all our test cases covering use cases in real life. Since the code needed a lot of tweaking before final, this became the start of another iterative process. At this stage, we also

tried to find the respective placements of the probes. Since the Phone application is constantly affected by changes, the placement for some probes had to be reconsidered several times.

- Analyzing the design and implementation
- Push the code to Gerrit for reviewing
- Discuss the design and implementation

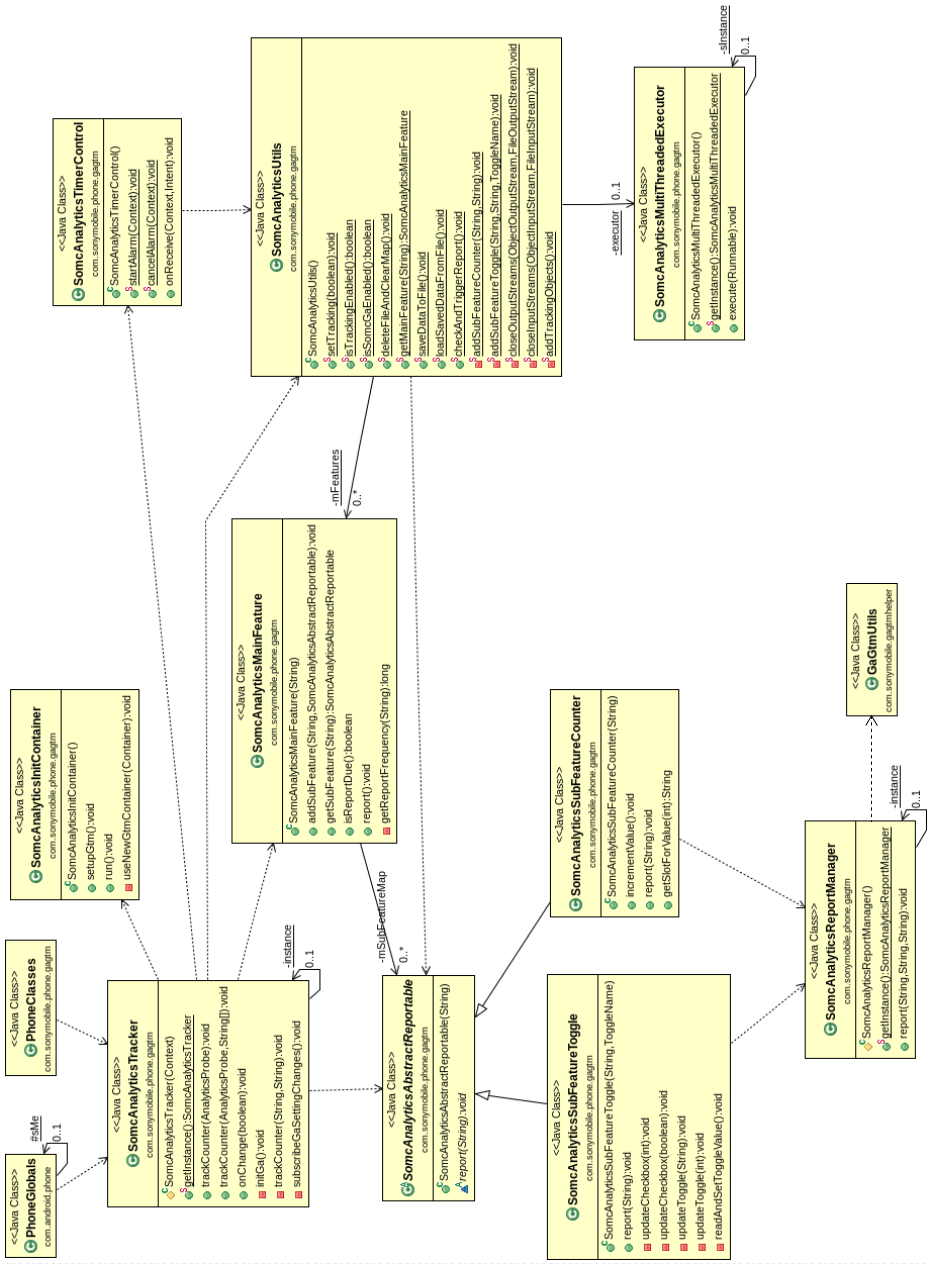


Figure 4.1: UML Diagram

4.4.12 Result of Our Implementation

A MainFeature object can represent a main feature i.e. "Call settings" in Settings. The SubFeature objects are used to represent the sub features of a main feature e.g. "Slow talk" and "Equalizer". Therefore, a MainFeature object can contain multiple SubFeature objects but a SubFeature belongs to only one MainFeature.

We have also defined two types of SubFeature objects, which we call Counter and Toggle. Counters are the SubFeature objects used when there's a need to keep track of the number of activations. So if we have a Counter object called "Answering machine" we will have a counter that keeps track of how many times "Answering machine" has been activated for an individual user. Toggle objects has been created to report which option that have been toggled for a setting such as a checkbox or a setting containing a list of options. Example of a checkbox is "Answering machine" which then can be "Enabled" or "Disabled", and an example of a setting with a list of options is "Equalizer" which can have the value "Normal", "Smooth" or "Bright".

Each MainFeature object has a Frequency obtained from the GTM container that indicates the period of how often it will report, I.e. if the MainFeature object "Call settings" has a Frequency of a week, all of its SubFeature objects will report their accumulated values once a week.

The values from GTM are updated twice every 12 hours automatically and can therefore be changed OTA without modifying the code itself.

To make the values more readable in the event reports in GA web interface, instead of reporting the exact value of a Counter object, the values will be categorized to a slot prior to reporting. The slots that we have defined are 0, 1, 2 5, 6-10, 11-15, 16-20, 21-40, 41-60 etc.

4.4.13 Functional Testing

For Usage Tracker in its early stages we concluded it would be more effective if we carried out the functional testing ourselves. These tests were done by trying out different scenarios and then comparing the outcome with the expected outcome in either ADB logcat or in the Event reports in GA web interface.

Testing was performed every time there was a modification on the code and each time the repository was synchronized with the latest version we had to recheck all probes and redo the testing.

The testing was to verify the following:

- Does the Counters count correct.
- Are the values reported to GA correct.
- Are the report frequencies correctly read from GTM.
- Does the MainFeatures containing their SubFeatures report simultaniously.
- Are the names of the main features and sub features correctly presented in GA.
- Does the Custom Dimensions work as intended.

- Does it work to change values OTA through GTM.
- Does the Toggle objects report the correct values.
- Does the check for report trigger at boot up and after each given interval.
- Does the GTM container update work as expected.
- Does save to file and read from file work.

Chapter 5

Presentation of Data

5.1 How Event Reports Are Normally Presented in GA

An event consists of four components and is used to describe a user's interaction with the application's content. These four components can be seen in Figure 5.1 below.

The values set for the components will be displayed in the event reports in the GA web interface. By having these values well organized data can be easily located and interpreted in the event reports.

These four components are sorted in an explorer type model in GA, with the Category label at the top of the hierarchy. Category should therefore be the main identifier to sort the events for a report. It can e.g. be button, video, download. Action is the view next in line after Category and is used to describe a particular Category. Action is also defined by a string and can be arbitrarily set to best describe the action. In the case of Category being Video, potential values of Action would be Play, Pause and Stop. For the later components Label and Value, these are not mandatory and can be used to give further information.

I.e. if we want to measure every time "Slow talk" in "Call settings" have been clicked we could e.g. send an event with the following values for the components: Buttons for Category, with the Action as click and the Label would be "Slow talk". Below is an example.

Field Name	Tracker Field	Type	Required	Description
Category	`Fields.EVENT_CATEGORY`	`String`	Yes	The event category
Action	`Fields.EVENT_ACTION`	`String`	Yes	The event action
Label	`Fields.EVENT_LABEL`	`String`	No	The event label
Value	`Fields.EVENT_VALUE`	`Long`	No	The event value

Figure 5.1: Event Tracking

5.2 How We Are Presenting Event Reports

To make the data more readable in GA we're presenting the data differently from the orthodox way. Thus, the values for the four components of an event will be set differently. For easy access to data concerning each task, we have chosen to categorize all information for a given task under its task name. I.e. all sub features to the main feature "Call settings" in "phone Settings" will be grouped together in the event reports under the common name "Call settings". This allows for fast access to all data concerning the task "Call settings". To do so all information that we're reporting to GA that concerns e.g. "Call settings" will have "Call settings" as the value for "Category".
Figure 5.2 below presents an example of a list with main features.

Event Category	Total Events
	7,661 % of Total: 100.00% (7,661)
1. Call Settings	3,535 (46.14%)
2. AM Settings	751 (9.80%)
3. Bluetooth Headset Audio Routing	575 (7.51%)
4. After Call Screen	544 (7.10%)
5. Voicemail Settings	489 (6.38%)
6. Wired Headset Audio Routing	474 (6.19%)
7. In Call Button Panel	438 (5.72%)
8. Route Sound	408 (5.33%)
9. Answering machine - from where is it set up	192 (2.51%)
10. Answering Machine - manual or automatic launch	185 (2.41%)

Figure 5.2: List of Main Features

To be able to distinguish between the different information reported to GA under each "Category" we'll make use of the "Action" parameter. The value for this parameter can e.g. be the name of the options belonging to the main feature. In the case of the "Category" being "Call settings", the value for the "Action" parameter could e.g. be "Slow talk", "Increasing ringtone" or "Equalizer".
Figure 5.3 below illustrates an example of the main feature "Call settings" with its sub features.

12. TTY mode	42 (4.12%)
13. Microphone noise suppression	33 (3.24%)
14. Equalizer	32 (3.14%)
15. slow talk	32 (3.14%)
16. Vibrate when ringing Toggle	30 (2.94%)
17. Dialpad touch tones Toggle	22 (2.16%)
18. Equalizer Toggle	22 (2.16%)
19. Increasing ringtone Toggle	22 (2.16%)
20. Microphone noise suppression Toggle	22 (2.16%)
21. Slow talk Toggle	22 (2.16%)
22. TTY mode Toggle	22 (2.16%)

Figure 5.3: List of Sub Features for the Main Feature Call Settings

The above two parameters "Category" and "Action" are compulsory out of the four. For our purposes, we'll need to utilize the third one called "Label".
With our Usage Tracker implementation, we will utilize the "Label" parameter to describe the value for the "Action" parameter. In the case of "Action", being a button the "Label" will then be a string value in the form of an interval e.g. 11-15. This tells us that the button have been clicked 11-15 times by the individual user in the specified time interval. In Figure 5.4 below we can see an example of this.

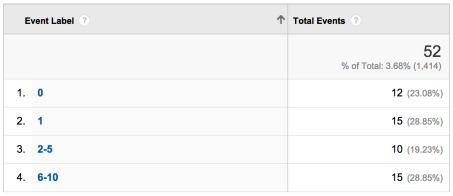


Figure 5.4: List Presenting the Number of Times a Specific Content Has Been Activated

If the "Action" is of the type Checkbox then the "Label" parameter can have the value either "Enabled" or "Disabled". This tells us if the checkbox was toggled or not prior to the report to GA. An example of this is shown in Figure 5.5 below.

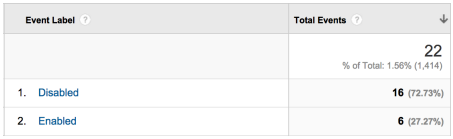


Figure 5.5: List Presenting the Values for a Specific Checkbox

If the "Action" is a setting with a list of options that can be selected. I.e. "Equalizer" with the selectable options Normal, Bright and Smooth. The value for the "Label" parameter will be the option selected by the user prior to the report to GA. An example is shown in Figure 5.6 below.

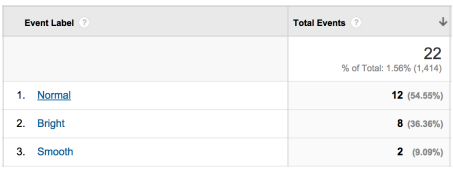


Figure 5.6: List Presenting the Values for Equalizer

5.3 Custom Reports

By using a custom report, we can filter the statistics in the way we want to and save a "shortcut" for these filtered values. We can for example filter the data for all the users in Sweden that uses Tele2. That custom report will only include the data for those specific users. Every time we want to see the statistics for those users, we can preferably select the custom report instead of filtering the statistics each time. To give an example we have generated random data and created a custom dimension that gives us the possibility to filter the statistics for all users in a country that uses a specific operator. In this custom dimension, the first step is to select the country of interest. Figure 5.7 tells us that we have generated statistics from users in three different countries, Finland, Sweden and Norway.

Device Sim Mcc	Total Events ↓
	1,020 % of Total: 72.14% (1,414)
1. Finland	534 (52.35%)
2. Sweden	471 (46.18%)
3. Norway	15 (1.47%)

Figure 5.7: First Step in the Custom Report, Select a Country

Next step in the custom report is to select the operator of interest. In this example, Sweden was selected in previous step and a list of all operators that the users are using will appear. Figure 5.8 presents statistics for users in Sweden that has Tele2 or Telia as an operator.

Device Sim Mnc	Total Events ↓
	471 % of Total: 33.31% (1,414)
1. Tele2	265 (56.26%)
2. Telia	206 (43.74%)

Figure 5.8: Second Step in the Custom Dimension, Select an Operator

After selecting the operator, i.e. Tele2 a list of all the main features will appear as in the following example. The statistics for each main feature corresponds to data accumulated from people using Tele2 as an operator in Sweden. See Figure 5.9 for reference.

Event Category	Total Events
	7,661 % of Total: 100.00% (7,661)
1. Call Settings	3,535 (46.14%)
2. AM Settings	751 (9.80%)
3. Bluetooth Headset Audio Routing	575 (7.51%)
4. After Call Screen	544 (7.10%)
5. Voicemail Settings	489 (6.38%)
6. Wired Headset Audio Routing	474 (6.19%)
7. In Call Button Panel	438 (5.72%)
8. Route Sound	408 (5.33%)
9. Answering machine - from where is it set up	192 (2.51%)
10. Answering Machine - manual or automatic launch	185 (2.41%)

Figure 5.9: Third Step in the Custom Report, Select a Main Feature

To filter the statistics for a specific time interval to ensure that each event corresponds to one user. We need to filter after users with the same Frequency. Figure 5.10 tells us that we have generated data for two different frequencies for the main feature "Call settings".

Frequency (days)	Total Events
	168 % of Total: 11.88% (1,414)
1. 7	117 (69.64%)
2. 14	51 (30.36%)

Figure 5.10: Fourth Step in the Custom Report, Select a Frequency

After selecting the Frequency, we will get a list of all the Sub features which can be seen in Figure 5.11. This list now only contains data from users in Sweden that has Tele2 as operator.

12. TTY mode	42 (4.12%)
13. Microphone noise suppression	33 (3.24%)
14. Equalizer	32 (3.14%)
15. slow talk	32 (3.14%)
16. Vibrate when ringing Toggle	30 (2.94%)
17. Dialpad touch tones Toggle	22 (2.16%)
18. Equalizer Toggle	22 (2.16%)
19. Increasing ringtone Toggle	22 (2.16%)
20. Microphone noise suppression Toggle	22 (2.16%)
21. Slow talk Toggle	22 (2.16%)
22. TTY mode Toggle	22 (2.16%)

Figure 5.11: Last Step in the Custom Report, Select a Sub Feature

5.4 Different Views for the Statistics

In the GA web interface it is possible to switch the presentation of the data by clicking on the buttons that can be seen in Figure 5.12.

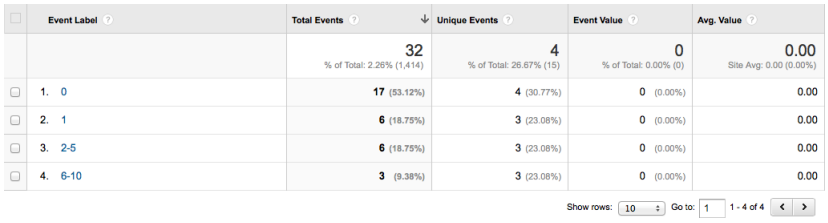
Figure 5.12: Different Views of the Data

From left to right, these buttons presents the data in the following way:

- Table view
- Percentage view
- Performance view
- Comparison view
- Pivot view

5.4.1 Table View

Table view organizes the data in a spreadsheet manner and is the most common view when going over statistics. Figure 5.13 illustrates an example.



Event Label	Total Events ↓	Unique Events	Event Value	Avg. Value
	32 % of Total: 2.26% (1,414)	4 % of Total: 26.67% (15)	0 % of Total: 0.00% (0)	0.00 Site Avg: 0.00 (0.00%)
1. 0	17 (53.12%)	4 (30.77%)	0 (0.00%)	0.00
2. 1	6 (18.75%)	3 (23.08%)	0 (0.00%)	0.00
3. 2-5	6 (18.75%)	3 (23.08%)	0 (0.00%)	0.00
4. 6-10	3 (9.38%)	3 (23.08%)	0 (0.00%)	0.00

Show rows: 10 Go to: 1 1 - 4 of 4

Figure 5.13: Table View of the Data

5.4.2 Percentage View

Effective way to get an quick overview of the proportions of the values. Figure 5.14 illustrates an example.

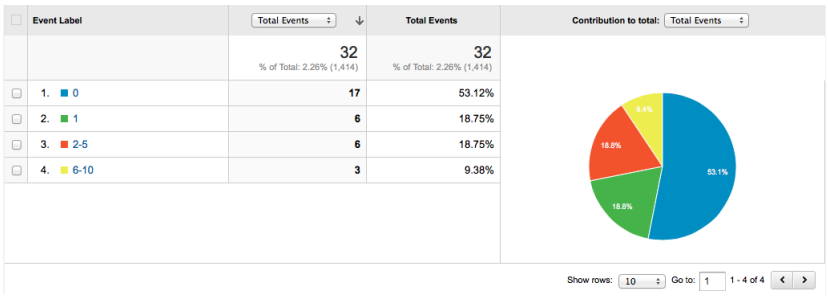


Figure 5.14: Percentage View of the Data

5.4.3 Performance View

The Performance view is also a tool for visualizing data and works as a complement to Percentage view. Figure 5.15 illustrates an example.

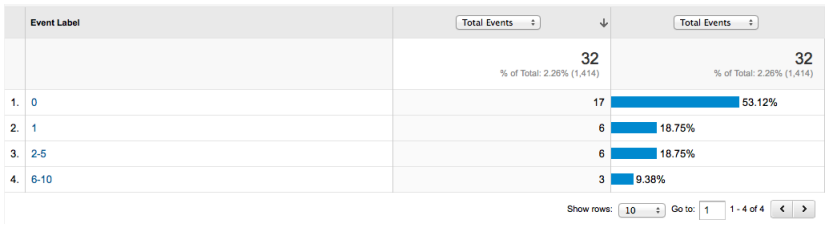


Figure 5.15: Performance View of the Data

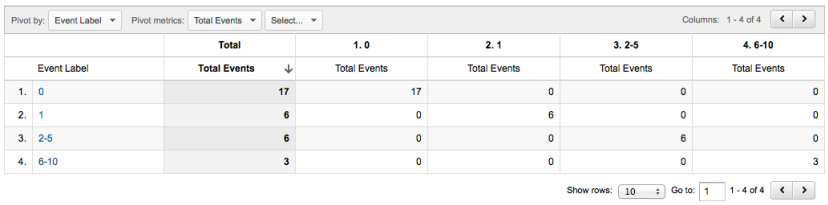
5.4.4 Comparison View

The comparison view is useful for comparing data distribution against each other. Figure 5.16 illustrates an example.

Figure 5.16: Comparison View of the Data

5.4.5 Pivot View

The pivot view is used when there is a need of viewing multiple dimensions of data at once. Figure 5.17 illustrates an example.



		Total	1. 0	2. 1	3. 2-5	4. 6-10
	Event Label	Total Events ↓	Total Events	Total Events	Total Events	Total Events
1.	0	17	17	0	0	0
2.	1	6	0	6	0	0
3.	2-5	6	0	0	6	0
4.	6-10	3	0	0	0	3

Figure 5.17: Pivot View of the Data

Chapter 6
The Telephony Code

The Telephony code is rather large and consists of many classes. Since our stay at Sony Mobile is very limited, we couldn't afford to spend much time into getting accustomed with the code. Instead, we used different techniques to navigate us to the right place to insert the Usage Tracker probes.

6.1 Work Flow for Inserting a Google Analytics Probe

To do this effectively we followed this work model:

1. Logs
2. Opengrok
3. verification

6.1.1 Logs

Having a quick look over some classes we recognized that the code was well logged. The following are different types of logs that could be found : [12].

- Verbose
- Debug
- Information
- Warning
- Error

By connecting the phone to the computer, the logs could be read in real time with ADB using the commando "Logcat". Filtering the outputs after certain keywords and log type, we could capture content relevant for our search, such as class name and method name etc. The result could then be investigated using OpenGrok.

6.1.2 Opengrok

OpenGrok is a source code search that Sony Mobile employees have access to. All code from Sony Mobile can be found using OpenGrok and it allows for quick search and has advanced filtering options. Using Opengrok we can list all the classes that contains specific search parameters. This was done for the keywords that we found interesting from the outputted ADB logs. Typically there are several places where a probe can be inserted and still generate the same result. The most appropriate place in the code would be the execution place for what the probe is actually measuring. E.g. if we want to measure the number of times button "A" is being pressed, the best place to put the probe is in the method which handles this button click. Usually it's verify straight forward but sometimes additional code needs to be added to accommodate the probe.

6.1.3 Verification

The last step is to verify that the new addition of probes works as expected by comparing the outcome and the results from looking at the GA logs in ADB logcat and output in the Event reports in GA web interface.

6.2 Sony Mobile Internal Testing

Prior to a feature being released, it has to go through various types of testing to ensure that all quality aspects are met.

6.2.1 Internal Testing in Sony Mobile

Prior to developing Usage Tracker the utilization of GA would be considered as a modification of another feature, which means that it wouldn't need to go through all of the testing. Since Usage Tracker had to be developed, the addition of it consisted of more code than expected which resulted in Usage Tracker being a feature itself. The result of this is that all code that plausibly were affected by our modifications had to go through regression testing and Usage Tracker itself had to undergo the following tests:

- Regression testing
- Functional testing
- Performance testing

Regression Testing
The intention of regression testing is to confirm that changes in the code hasn't introduced new problems to prior working code. This is usually done by rerunning previously completed tests.

Functional Testing
Functional testing is performed to verify that everything is working as expected.

These tests performed by Sony Mobile covered all our private tests while also covering other specifications of the phone. The following additions of tests were specifically added for our interest:

- Counters updates their values correctly
- Periodic update of results to GA
- Enable/Disable of feature
- Enable/Disable flight mode

Performance Testing
Performance testing tests among other things a systems responsiveness and stability. It's important to check if the addition of Usage Tracker had any impact on these factors. Since Usage Tracker saves a file at shutdown and reads from a file at boot up, extra care was done when performing tests for measuring boot up time and shutdown time. Testing showed that the addition of Usage Tracker didn't slow down the phone noticeably for the user.

6.3 Improvements

The implementation of Usage Tracker could've been more efficiently, the most obvious is that all Counter objects for the sub features are reported once each time period, even the Counter objects for the sub features that has not been activated at all will be reported.

This should've been avoided to save data traffic and unnecessary operations as the following.

- Reporting these Counter objects.
- When starting the phone, reading these Counter objects from a file.
- In reboot, writing these Counter objects to a file.

This means that we only want to process Counter objects that has been activated at least once, but retain information about the Counter objects that has not been activated. In retrospect, we came up with the following solution.

Instead of reporting all Counter objects, it's more efficient to only report the Counter objects that has been activated at least once. No other Counter objects that has not been activated will be reported, this means that we'll never report a value of "0". Nevertheless, we are still interested in the Counter objects that has the value "0" since we want to know how many users that are not using a certain feature.

This can be achieved by introducing a new Counter object for each main feature that will always be reported by all the users. The objective for this Counter object is to record the number of total users. A better explanation will be provided by following example. In this example the newly introduced Counter object is called "Total users". Table 6.1 presents how the Sub features for a specific main feature are grouped in our current solution.

Table 6.1: Example of How Sub Features for a Specific Main Feature Are Grouped in the Current Solution

Sub feature	Total events
Sub feature 1	50
Sub feature 2	50

With the current solution all sub features belonging to a common main feature are reported exactly the same number of times. This is because each user will always report their Counter objects, whether a Counter object has been activated or not. Selecting Sub feature 1. The Statistics may then look like Table 6.2 below.

Table 6.2: Example of How Counter Values May Look Like in the Current Solution

Counter value	Events
0	40
1	5
2-5	5

This gives us that 40 out of 50 users have reported that this Sub feature has not been activated at all during the time interval. If we have 60 Sub features for this main feature that have similar statistics as above, with 40 out of 50 users that haven't activated the Sub feature at all, then this will result in 40 users each reporting 60 Counters with the value "0" which can be omitted with the new design. The new design makes it possible to report a single Counter object that represents all these 60 Counter objects which reports the value "0".

Table 6.3 presents the list of all main features. This design has been consistent from the old to new solution.

Table 6.3: List of All Main Features Is Consistent from the Old to New Solution

main feature	Total events
Feature 1	100
Feature 2	200

In Table 6.4 is an example of how the sub features will be presented in the new solution with the introduction of "Total users" which gives the number of total users that are reporting.

Table 6.4: Example of How Sub Features Are Presented in the New Design

Sub feature	**Total events**
Total users	50
Sub feature 1	10
Sub feature 2	20

"Total users" informs that there are a total of 50 users who are reporting statistics, and we can also see that only 10 users has reported for Sub Feature 1 and 20 users has reported for Sub Feature 2. This means that 40 users has not activated sub feature 1 and 30 users has not activated sub feature 2.
Selecting e.g. Sub feature 1 will result in the following presentation in Table 6.5.

Table 6.5: Example of How the Counter Objects Values Will Be Presented in the New Design

Counter value	**Events**
1	5
2-5	5

The value "0" haven't been reported but we can still derive that information with the help of "Total users" and the number of events that we have received. Calculations gives us 50 - 10 = 40 meaning that 40 users haven't used the sub feature.

The other positive thing with this solution is that you do not have to create all the Counter objects. We can instead create one "Total users" Counter object for each main feature, the rest of the Counter objects will be created when they are activated the first time.

This solution is more optimized and saves on data traffic but makes the presentation of data less intuitive.

Chapter 7

Conclusions & Discussion

7.1 The Thesis Work

The scope of the thesis work were the following:

- Investigate and suggest a strategy for how to utilize GA.
- Evaluate how to present the data in GA.
- Implement a selection of GA probes.
- Avoid unnecessary patching of Android.

By presenting statistics of concrete data, Usage Tracker helps with answering questions that we previously only could speculate about. With the development of Usage Tracker and the utilization of it we managed to fulfill all the requirements of the thesis proposal.

At the beginning of this thesis work, our impression of the implementation of GA was that it would be very straightforward. We tried to confirm this early on by creating a prototype that we implemented GA to. Even though we didn't have any prior experience of GA it was easy to use. Therefore, our first impression was that the weight of this thesis work would primarily lay on investigating on how to effectively utilize GA and how to present the data.

Naturally we'd been optimistic of extending the scope of our thesis work with ambitions to release the code worldwide to collect real usage data and then investigate it. But after receiving the task list from our scope owner, we realized that we had to develop a complementary tool, Usage Tracker. This came to be more time consuming than expected.

Studying in University, we are used to take quality aspects in consideration when developing software programs, however with the ambition to release our code worldwide, the quality aspects of our thesis work were of a much greater extent than we were used to. This to minimize any potential problems that could occur and potentially render millions of people unable to use their phones to their extent.

The unforeseen development of Usage Tracker had a big impact on our time plan having us miss our milestones, which made it less likely for us to experience the release of our code worldwide. We came to realize that we wouldn't make our

deadline but made the decision to persist and extend our stay to get our code ready for integration into a future release.

In the end the design of Usage Tracker got approved and it passed all the tests. Our thesis work was then ended with a presentation of our work.

Sony Mobile is gradually applying a more data driven approach when forming their decisions on features and we are pleased to hear that our Usage Tracker will be integrated into a near future release.

Applying Usage Tracker in parallel with the internal usability testing allows us to obtain both heaps of quantitative data along with qualitative data from the users. However, since "Usability tests are best for observing behaviors and measuring performance issues, while perhaps gathering some qualitative information along the way" we need a third complement to gather quality data about people's opinions and attitude about an preliminary concept. Focus group research would in this case be appropriate [5].

Focus group research has the advantage to explore a few people's opinions and attitude towards a preliminary concept in greater depths. The participants are free to share their thoughts with each other and discuss them. These thoughts are usually different from what they actually do and is therefore very different from the results received from usability testing. [5].

7.2 Future Work

If priority lies within prioritizing features then it's more benefitting to use the current Usage Tracker or parts of it to collect real statistics from end users. The statistics can then be compared to the statistics from the internal usability tests to verify if there are any correlations. Matching results would be a strong implication to take appropriate action to the sub feature. Otherwise, if the statistics are contradictory to each other an investigation could take place for further examination.

For Usage Tracker, the most obvious work would be to fix the drawbacks presented. It would also be helpful to add a functionality that displays the correlation between the usage of different sub features. I.e. people that uses sub feature A are they also more keen to use sub feature B?

Another improvement, if possible, would be to query raw data to GA and then process it instead of processing the data prior to querying it to GA. This would retain all the information of the data and the functionality in Event reports in GA web interface could be used more efficiently.

References

[1] Mobile App Analytics,
http://www.google.com/analytics/mobile/

[2] Mobile App Analytics,
https://www.google.se/tagmanager/features.html

[3] GA platform overview,
https://developers.google.com/analytics/devguides/platform/

[4] ISO 9241-11,
https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en

[5] Rubin, J and Dana, C. (2008) *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2^{nd} Edition, Indianapolis: Wiley

[6] Victoria Wibeck. (2010) *Fokusgrupper: Om fokuserade gruppintervjuer som undersökningsmetod*, 2^{nd} Edition, Lund: Studentlitteratur

[7] George, E.P Box., Stuart Hunter, J. and William G, H. (2012) *Praktisk statistik och försöksplanering* Lund: Studentlitteratur

[8] Tullis, T and Albert, B. (2008) *Measuring the user experience: Collecting, Analyzing and Presenting Usability Metrics*, Amsterdam ; Boston: Morgan Kaufmann

[9] Shneiderman, B and Plaisant, C. (2009) *Designing the user interface: Strategies for Effective Human-Computer Interaction*, 5^{th} Edition, Harlow: Pearson Education

[10] Soren Lauesen (2002) *Software Requirements: Styles and Techniques*, Harlow: Addison-Wesley

[11] Chaunsey Wilson (2013) *Brainstorming and beyond: a user-centered design method*, Burlington, Mass. : Morgan-Kaufmann

[12] Android Logs,
http://developer.android.com/tools/debugging/debugging-log.html

[13] Android service definition,
http://developer.android.com/guide/components/services.html

[14] Event tracking in GA,
https://support.google.com/analytics/answer/1033068?hl=en