Deliverable D4.5
Development status of smartphone technology for track and ride quality monitoring
Submission date: 27/05/2016
Lead contractor

ADS (ADS-ELECTRONIC RESEARCH)

Contributors

TU Delft, AFER, RCCF

Project Coordinator

University of Sheffield, USFD
Executive Summary

One of the objectives of Task 4.3 is the development of an application for Low Cost Smartphones, to estimate ride comfort and to obtain track inputs (e.g. vibration, speed, indoor temperature and rail unit GPS positions etc.). The application is:

- based on smartphone accelerometer and GPS receiver;
- designed for data collection in order to measure ride comfort and track inputs.

The general goal of this task is the development of a smartphone based technology for vehicle and infrastructure monitoring from within passenger vehicles, i.e. crowd-sourced data collection, to increase the regularity and granularity of the monitoring data available.

Expected results of this task are:

(i) Developing an app to gather data from the smartphone GPS sensor and its accelerometer. This will consider conservation of battery life as a priority to ensure viability of the app.
(ii) Developing a gateway to which the data is transmitted using the phone 3G or WiFi connection.
(iii) Developing an interface for querying of available data (e.g. relational database structure).

Research activities carried out under phase 1 of the Task 4.3 focused on the overall design of the Low Cost Smartphone system and the design of each local component to create a dynamic Web services and mobile platform support for data collection in order to measure ride comfort and to obtain the track inputs.

Design activity has mainly focused on the following actions:

- Design of the logical processing components (business logic) of the mobile terminal application;
- Design of the user interface for mobile terminal application;
- Design of the system Web services;
- Design of the database of the system;
- Design of the logical processing components (business logic) of the reporting web interface.

Functional and technical specifications and system architecture are developed by two sub activities namely:

- Design of the specifications involving demarcation of functional and technical requirements of the system in concrete terms (functions, processing, interfaces, etc.);
- Design of the system architecture that includes the reference model, data model consolidated, logical and physical architecture and design of the system warehouse.
Table of contents

Executive Summary ................................................................................................................................................ 3
Table of contents .................................................................................................................................................... 4
Abbreviations and acronyms .................................................................................................................................. 5
1 Introduction ................................................................................................................................................... 6
  1.1 Working Methodology......................................................................................................................... 6
  1.2 Technical Approach ............................................................................................................................. 6
2 Requirements for system for track and ride quality monitoring .............................................................. 8
  2.1 Requirements for Android application .............................................................................................. 9
  2.2 Requirements for crowd-data server and BI application .................................................................... 9
  2.3 Requirements for quality attributes of systems .................................................................................. 9
3 Purposes of system for track and ride quality monitoring based on Low Cost Smartphone .................... 12
  3.1 Android architecture and Sensor architecture in Android ................................................................ 13
  3.2 Design of Smartphone system ........................................................................................................... 16
    3.2.1 Design of Android application service for sensor acquisition data ...................................... 16
      3.2.1.1 Use Case Diagram .................................................................................................................. 17
      3.2.1.2 Class Diagram ......................................................................................................................... 17
      3.2.1.3 Sequence Diagram ................................................................................................................ 18
      3.2.1.4 Smartphone database ......................................................................................................... 19
      3.2.1.5 Service for collecting data provided by Accelerometer and GPS ....................................... 22
    3.2.2 Design of Android application gateway for data transmission .............................................. 28
    3.2.3 Design of Android application for local reporting and interrogation data ............................. 31
  3.3 Design of crowd-data system .............................................................................................................. 32
    3.3.1 Design of RESTful service & common database for crowd-data server ................................ 34
    3.3.2 Design of reporting web interface for crowd-data server ......................................................... 36
4 Strategies for creating an accessible and useful system for track and ride quality monitoring based on Low Cost Smartphone ................................................................. 37
5 Conclusions .................................................................................................................................................. 39
6 References ................................................................................................................................................... 40
### Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation / Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT</td>
<td>Android Development Tools</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface - is a set of routines, protocols, and tools for building software applications</td>
</tr>
<tr>
<td>BI</td>
<td>Business intelligence - a set of techniques and tools for the acquisition and transformation of raw data into meaningful and useful information for business analysis purposes.</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HAL</td>
<td>HAL (Hardware Abstraction Layer) - defines a standard interface for hardware vendors to implement and allows Android to be agnostic about lower-level driver implementations.</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>LBS</td>
<td>Location-based service (LBS), are made up of a set of applications that exploit the knowledge of the geographical position of a mobile device (smartphone) in order to provide services based on that information.</td>
</tr>
<tr>
<td>RESTful</td>
<td>Representational state transfer (REST). REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system.</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

This report represents the deliverable: “Development status of smartphone technology for track and ride quality monitoring”.

The objectives of the deliverable are to describe the following:

- Functional specification that provide a general description of the system including the actors, system functions, data model consolidated and database structure;
- Reference architecture;
- Logical and physical architecture;
- Principal user interfaces;
- Mobile terminal facilities that will be used in software development;

1.1 Working Methodology

For research activities performed within the first technical stage of the Task 4.3, the research team used the following working methodology.

- Evaluation existing techniques and solutions

The work started with the evaluation of technologies, tools, algorithms, solutions and techniques used in mobile terminal application and respectively of the Web application design, and also of those needed for measuring the acceleration, the force of gravity and rotation by sensors.

- The system design

The design took into account the results of the performed assessment step and sought to improve system performance by the defined configuration solution and to consider the best algorithms for use in various operational scenarios. The design was based on the specification of technologies and techniques for each type of system functionality.

- Risk Assessment

A high-level design of an alternative, if unexpected obstacles arise in the initial technical design (such as failure in implementation, inadequate results, etc.) was taken into account when they were selected to use technological solutions (e.g. the possibility uses an external GPS, see section 4) to achieve the system.

Important decision points of passage or not (Go, No-Go milestones) from one activity / stage to another will facilitate decision making and the possible switch to an alternative technique.

1.2 Technical Approach

- System lifecycle

Lifecycle (see Figure 1-1) of system consists of the following steps:

1. Requirements - Gathering of the requirements
2. Specification - Define functional requirements of the system;
3. Design - System design;
4. Implementation - implementation of the system;
5. Integration - Integration of the components and verification of integrated system;
6. Maintenance - Maintenance of the system, removal of any abnormal functionalities and formulating of the development needs;
7. Update requirements - Update system requirements and launch a process of its development.

Aspects considered

The design stage of the application for mobile terminals and Web services of system focuses on the following aspects:

- The first is to consider the requirements;
- Development of the functional specification which assume the delimitation of functional requirements in concrete terms namely, internal processing, interfaces that are able to meet all requirements;
- Designing of the system architecture which will include reference model, data model consolidated logical and physical architecture;
- Designing and specification of the implementation in detail for each component, including objects and methods to be implemented and interfaces to be provided:
  - Designing of the system database;
  - Designing of the logical processing components of system;
  - Designing of the logical processing component of the application for mobile terminals;
  - Designing of the user interface of the application for mobile terminals.
  - Designing of the user interface of crowd data system;
2 Requirements for system for track and ride quality monitoring

The main goal of T4.3 is development of an application for a Low Cost Smartphone:

- Using Android SDK.
- Based on smartphone accelerometer and GPS (global positioning system).
- For data collection in order to measure ride comfort and obtain the track inputs through a low-cost approach which will collect large volumes of data.

Software processing will extract the underlying information in order to:

- extend the availability of good quality asset management data,
- supporting improved management and lean operation.

The global system architecture is presented in the Figure 2-1 underlying the different physical tiers (system applications) and connection points.

At rail unit level (Application for low cost smartphone) different physical tiers of system consist of:

- Android application service for sensor acquisition data.
- Android application gateway for data transmission (Adhoc OR Automatic by Scheduler) via RESTful-service using the phone 3G/4G or WiFi connection.
- Basic Android application for local reporting and interrogation data.
At Control Centre (Application for crowd-data server) different physical tiers of system consist of:

- Web application for crowd-data server (RESTful service & common database).
- BI application (reporting web interface) for crowd-data server.

In the design of our system shall be considered features of existing appropriate systems (e.g. SUMO system developed by RNCF, system mentioned in User group meeting in Paris), features that will be improved and extrapolated to a crowd-data system.

### 2.1 Requirements for Android application

The application for mobile terminals will operate at the Rail Unit level, and consider the following requirements and functional constraints:

- It will carry out an application that will use components included in smartphone, for data acquisition: geographical position and intensity of the vibrations during the trip.
- This application includes some risks because of not expecting high precision measurements from a consumer and commercial low cost device; in addition, this acquisition device will be placed after suspensions and mechanical fastening systems of the wagon wheels.
- Another important aspect is the increasing autonomy of operation of measuring mobile platform: Smartphone. In this aspect, it will be completed with an external high capacity accumulator connected via micro USB interface. From the viewpoint of software application, to reduce consumption, all applications and interfaces which have no function in activities dedicated to achieving good quality measurements will be closed.
- To achieve a 3G/4G or WiFi gateway will be programmed a communication module type Android Service which will manage data transmission, remotely to a Control Centre. This module will run in the background and will listen to requests for transmitting information, the amount of data requested will be sent on the same communications support that was received: GSM or WiFi.
- The module type like Service class will be launched automatically when the acquisition application from sensors starts but, also can be independently turned on from the device list of Tablet / Smartphone.

### 2.2 Requirements for crowd-data server and BI application

The application for crowd-data server will operate at the Control Centre level, and consider the following requirements and functional constraints:

- Data is transmitted via the communication module described above, an application will be made that will contain the following functional components: data communication, processing and storing data received.
- User interface will provide functions through dedicated graphical controls for:
  - database query
  - display information in analysis format
  - commands and queries toward the mobile device, etc.

### 2.3 Requirements for quality attributes of systems

The design of system will be focused on the quality attributes thinking around critical problems that the design should solve.
The quality attributes are addressed during design rather than during development in order to improve the likelihood that the system will be successful in the long term.

The quality attributes considered in the design of the system are presented below:

- **Availability** - defines the proportion of time that the system is functional and is measured as a percentage of the total system downtime over a predefined period. Availability is affected by system errors, infrastructure problems, malicious attacks, and system load.

- **Conceptual Integrity** - defines the consistency and coherence of the overall design and includes the way that components are designed, as well as factors such as coding style and variable naming.

- **Flexibility** - system flexibility is the ability to adapt to varying environments and situations, and to cope with changes to business policies and rules. It is easy to reconfigure or adapt in response to different user requirements.

- **Interoperability** - System Interoperability is the ability of its diverse components to operate successfully by exchanging information by using services. Communication protocols, interfaces, and data formats are the key considerations for interoperability.

- **Maintainability** - System maintainability is the ability to undergo changes to its components, services, features, and interfaces as may be required when adding or changing the functionality, fixing errors, and meeting new business requirements. Maintainability can be measured in terms of the time it takes to restore the system to its operational status following a failure or removal from operation for upgrading. Improving system maintainability will increase efficiency and reduce run-time defects.

- **Manageability** - System manageability defines how easy it is to manage the application, through tools used in monitoring system and for debugging and performance tuning.

- **Performance** - is an indication of the responsiveness of the system to execute any action within a given interval of time. It can be measured in terms of latency or throughput. Latency is the time taken to respond to any event. Throughput is the number of events that take place within a given amount of time. Factors affecting system performance include the demand for a specific action and the system’s response to the demand.

- **Reliability** - System reliability is the ability of the system to remain operational over time and is measured as the probability that the system will not fail to perform its intended functions over a specified interval of time. Improving the reliability of the system may lead to a more secure system because it helps to prevent the types of failures that a malicious user may exploit.

- **Scalability** - System scalability is the ability to function well when there are changes to the load or demand. The system will be able to be extended by scaling up the performance of the server. Typically, the system should be able to handle increases in size or volume. The aim is to maintain the system’s availability, reliability, and performance even when the load increases.

- **Security** - System security is the way that the system is protected from disclosure or loss of information and prevents unauthorized modification of information. The factors affecting system security are confidentiality, integrity, and availability. Authentication, encryption, and auditing and logging are the features used for securing systems.

- **Supportability** - System supportability defines how easy it is for operators, developers, and users to understand and use the system, and how easy it is to resolve errors when the
system fails to work correctly. Supportability is the ability to provide support to the system when it fails to work correctly.

- **Testability** - System testability is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met. Testability allows the isolation of the faults in the system in a timely and effective manner.

- **Usability** - System usability defines how well the system meets the requirements of the user by being intuitive, easy to localize and globalize, able to provide good access for disabled users and a good overall user experience.
3 Purposes of system for track and ride quality monitoring based on Low Cost Smartphone

In this task, from the technical point of view, we propose the implementation of location-based service for real-time vibration monitoring a railway unit in order to measurement of ride comfort and obtain the track inputs.

Location-based services (LBS), are made up of a set of applications that exploit the knowledge of the geographical position of a mobile device (smartphone) in order to provide services based on that information. LBS systems generally consist of the following software and hardware components: a service provider’s software application, a mobile network to transmit data and requests for service, a content provider to supply the end user with geo-specific information, a positioning component (GPS), and the end user’s mobile device.

The work in this first work period of this task is divided into the following stages.

1) Design a Location Based Service application on Android Operating System.
2) Design and interface of a vibration sensor circuit to the Android smart phone.
3) Design a server side application to receive the vibration information and store it in a Web GIS database.
4) Design application to display and serve the information using dynamic mapping technique.

One of the main purposes of the task is to use the low cost smartphone and by 90 percent in this category runs the Android operating system. Android platform is a smart mobile phone platform
launched by Google. Android provides the support of mobile map and Location Based Service. The most important aspects regarding the feature of Android smart phone is that, the phone’s hardware functionality is not only limited to the manufacturer’s interest, but also can be extended by the user. Therefore, external components such as sensors or other remotely sensed objects can be connected to a smartphone based on Android, and which exploit phone’s hardware give added value to services.

3.1 Android architecture and Sensor architecture in Android

Android is a package of software for mobile devices, including an operating system, middleware and core applications (see Figure 3-2). The Android SDK provides powerful tools and APIs necessary to develop applications on the Android platform using the Java programming language.

Android platform is of open system architecture, with versatile development and debugging environment, but also supports a variety of scalable user experience, which has optimized graphics systems, rich media support and a very powerful browser. It enables reuse and replacement of components and an efficient database support.

![Android platform architecture](image)

According to “Android Open Source Project” (1), Android system architecture consists of the following layer:

1. **OS Layer:** Android used the Linux Kernel at the OS layer. The kernel is also responsible for memory management, process management, security and drivers. This layer provides the abstraction between the underlying hardware and the software system.

2. **Middleware:** Above the Linux kernel are set of C/C++ system libraries. Some of the important libraries include ‘libc’ which is the standard C system library. This has been customized for supporting embedded Linux environment. There are media libraries
support popular audio, video and image formats. A ‘SQLite’ database engine for the applications, ‘SGL’ which is the 2D graphic engine, ‘Webkit’ which is the open source engine that powers the browser, etc.

3. Dalvik Virtual machine (DVM): Also, the middleware consists of the Android runtime which includes the customized Dalvik virtual machine (DVM). It executes files in ‘dex’ format which is dalvik executable format. DVM converts the classes compiled by java virtual machine into dex format with the help of a tool called ‘dx’. The Linux Kernel handles threading, process management, memory management and other related issues for the Dalvik Virtual Machine. In addition to the DVM, the runtime also includes the Core Java Libraries which provides all the java functionality.

4. Application Framework: The application framework consists of:
   - Content Providers – It enables one application to share its data with another application. For example, the details of an individual stored in the phone book application could be used by the email application.
   - Activity Manager - It is responsible for handling the life cycle of you Activity.
   - Notification Manager - It is responsible for displaying important alert messages.
   - Resource Manager - It is responsible making the resources like gif, png images or layout files used for the UI display available to your application.

5. Applications: The Android system includes some built-in core applications like the email client, browser, maps, calendar etc. These applications have been written using the Java programming language. Any user developed applications exist at this level and make use of the underlying functionalities of the core java libraries.

Android provides OS-level and framework-level support for sensor event gathering, fusion, and processing. This well-established sensor architecture makes Android a robust platform for sensing applications.

Android system supports three types of sensors:
   - The first type is motion sensors that measure the acceleration and rotation of a device, e.g., accelerometers and gyroscopes.
   - The second type is environmental sensors that give the information about the surrounding environment of a device, e.g., barometers, thermometers.
   - The third type is position sensors that provide positional information for a device, such as orientation. These include orientation sensors and magnetometers.
According to Yin Yan (2014) (2), Sensor architecture in Android (see Figure 3-3) involves four layers—the kernel, HAL (Hardware Abstraction Layer), SensorService, and SensorManager.

1. **Kernel**: Android uses Linux as the base kernel and most of the mechanisms that Android uses for sensor support come directly from Linux. Namely, Android uses Linux’s input subsystem (evdev) for sensor devices. The input subsystem is an abstraction layer between input devices (e.g. accelerometer, gyroscope, etc.) and input handlers. The input devices capture physical inputs and produce input events. For example, an accelerometer reading results in three input events, one for each coordinate. The input events then go through the input core and are dispatched to any subscribed handlers, which in turn make them available through the standard Unix file interface /dev/input/eventX. Thus, applications can access the raw sensor data as an input event by calling POSIX system calls such as open(), read(), poll(), etc.

2. **HAL** (Hardware Abstraction Layer), which is a user-space layer that interfaces with the kernel. It polls the input events from kernel (/dev/input/eventX) for reading sensor data, and provides a unifying hardware interfaces for other user-space processes. This layer also hides vendor-specific details and hardware vendors must provide the actual implementation underneath the abstraction. This vendor-provided implementation essentially maps the HAL abstract interface to the device driver hardware interface for each sensor. HAL is loaded into a user-space process as a shared library.

3. **SensorService**, uses HAL to access raw sensor data. This layer is in fact part of a system process that starts from system boot time. The main job of this layer is two-fold. First, it reformats raw hardware sensor data using application-friendly data structures. Second, it fuses readings from multiple hardware sensors to generate software sensor data. Thus, this layer enables complete sensor access for both hardware and software sensors.

4. **SensorManager** is an Android library linked to each app at runtime. It provides registration and deregistration calls for app-implemented event handlers. Once an app registers an event handler, SensorManager’s SensorThread reads sensor data from a SensorService. A
SensorEventQueue is a buffer that holds sensor data pulled from the SensorService. The communication between SensorManager and SensorService takes two forms; an inter-process communication mechanism of Android called Binder, and a domain socket. Binder is used for communicating commands; a domain socket is used for data transfer.

From the technical point of view, application components are the essential building blocks of an Android application. There are following four main components that can be used within an Android application:

1. Activities - They dictate the UI and handle the user interaction to the smart phone screen;
2. Services - They handle background processing associated with an application;
3. Broadcast Receivers - They handle communication between Android OS and applications;
4. Content Providers - They handle data and database management issues.

### 3.2 Design of Smartphone system

When developing applications for the Android platform, the recommended and most popular integrated development environment (IDE) is IntelliJ IDEA or Android Studio (IDE developed by Google based on IntelliJ IDEA) with the Android Development Tools (ADT). The ADT extends IntelliJ IDEA with a graphical UI design tool, Android debugging, application testing on virtual and physical devices, and digital signing of the applications among other things.

The Android API uses XML files to define an application’s graphical layout separately from the code (Android Developers 2013e). It uses another specific XML file called a “manifest” to declare its version, components and application permissions within the system etc.

Code-wise, an application has four main components, namely Activities, Services, Broadcast receivers and Content providers (see Section 3.1).

In the next three sections are presented the design of the following Android applications:

- Android application service for sensor acquisition data
- Android application gateway for data transmission
- Android application for local reporting and interrogation data

#### 3.2.1 Design of Android application service for sensor acquisition data

This is the most important component of System at smartphone level, being responsible for the collection, storage and processing data.

From the physical point of view, this component consists of both the Activities, Services and Local Database.

The activities consist of UI processes responsible for setting and start / stop data acquisition services.
The service is responsible for collecting, processing GPS and sensors and also for writing them into database.

### 3.2.1.1 Use Case Diagram

Figure 3-4 shows the use case diagram for sensor acquisition application. In the use case diagram of the Sensor Acquisition, all the ovals represent the functionalities of the user (the user is represented by the rail unit from which collects data). The Add Data functionality helps the user to create and record data of interest based Location and Vibration Data. The Get My Location functionality locates the current location of the user with location latitude, longitude and altitude coordinates. The Get Vibration Data functionality helps the user to collect vibration data provided by the accelerometer sensor. View Saved Data functionality lists all the saved data (Location + Vibration) of the user, it helps the user to create specific log and share / send his saved log by data transmission services (see Section 3.2.2). The Settings functionality helps the user to Accelerometer setting characteristics, data connection for external GPS, modality to transmission data to the Server (Adhoc OR Automatic by Scheduler).

![Use Case Diagram](image)

*Figure 3-4 Application for sensor acquisition data Use Case Diagram*

### 3.2.1.2 Class Diagram

The class diagram consists of the interfaces, methods, variables and relationship between them. Figure 3-5 is the class diagram for Sensor Acquisition Data app which describes the major functionalities of the user like creating *Location / Time Collection*, *Vibration Collection* and saving them in the database.

A part from these functionalities, users can get the current location by using Get Location and can save the location coordinates in the database for to locate the place where the vibration occurred. The smartphone app database has three main tables, *Location table*, *Vibration table* and *Setting Value* table for storing and retrieving the data.
3.2.1.3 Sequence Diagram

A sequence diagram describes how the communication happens between the user (the user is represented by rail unit from which collects data), application, and local data base. Figure 3-6 describes the user sequence with the application and the application handling the user request. The application processes the user request and makes a request to the device’s local SQLite database to store and retrieve the data.
3.2.1.4 Smartphone database

SQLite is an open source SQL database that stores the database as a text file on a device, and also it is a relational database management system just like Oracle, MySQL and PostgreSQL.

Android has a built in SQLite implementation, and application specific database files are stored in a private disk space that’s inaccessible to other applications. This way, no application can access another application’s data.

For Application for sensor acquisition we just need three principal tables in the database. The three tables are:

- Settings - to store all setting variables related to rail unit and rail route
- Vibration - to store all vibration data provided by accelerometer sensor
- Location – to store location, speed and time data provided by GPS receiver
Application-level, model classes were created for each database table, except for StorageData table because it is a table associative. In next figure is presented source code for the model class for Vibration table.

Figure 3-7 Smartphone system Database structure

Figure 3-8 Source code for the model class for Vibration table
Database helper class contains all the methods to perform database operations like opening connection, closing connection, insert, update, read, delete and other things. As this class is helper class, place this under helper package. In next figure is presented source code for class **DatabaseHelper** for create and update Settings and Vibrations tables.

In this class, at each level of table were created these types operations:

- Create
- Update
- Delete
- Fetching a specific row
- Fetching a specific row filter by specific variables
- Fetching all

In the next sequence of source code (see Figure 3-10) is present procedure for reading all vibration filtered by a specific session.

![Figure 3-10 Source code for procedure for reading all vibration filtered by a specific session](image)

### 3.2.1.5 Service for collecting data provided by Accelerometer and GPS

This component of mobile system is an Android service application that collects sensor data from the micromechanical mechanisms (accelerometer and gyroscope) located within the mobile. Moreover, sensor data is synchronized with the mobile browser (URL) in real-time so that it can be stored (SQLite database, see section 3.2.1.4) for a posteriori analysis which involves applying multiple data mining techniques.

A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. According to Android API Guide (3) the service can essentially take two states:

- **Started** – A service is started when an application component, such as an activity, starts it by calling `startService()`. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
- **Bound** – A service is bound when an application component binds to it by calling `bindService()`. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).
A service has life cycle callback methods that you can implement to monitor changes in the service’s state and you can perform work at the appropriate stage.

The following diagram below on the left shows the life cycle when the service is created with `startService()` and the diagram below on the right shows the life cycle when the service is created with `bindService()`.

![The Android service lifecycle](source Android API Guide, Services) (3)

In order to create a service, it is necessary to create a Java class that extends the Service base class or one of its existing subclasses. The Service base class defines various callback methods and the most important are given below table.

<table>
<thead>
<tr>
<th>Callback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onStartCommand()</td>
<td>The system calls this method when another component, such as an activity, requests that the service be started, by calling <code>startService()</code>. If you implement this method, it is your responsibility to stop the service when its work is done, by calling <code>stopSelf()</code> or <code>stopService()</code> methods.</td>
</tr>
<tr>
<td>onBind()</td>
<td>The system calls this method when another component wants to bind with the service by calling <code>bindService()</code>. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an <code>IBinder</code> object. You must always implement this method, but if you don’t want to allow binding, then you should return null.</td>
</tr>
<tr>
<td>onUnbind()</td>
<td>The system calls this method when all clients have disconnected from a particular interface published by the service.</td>
</tr>
<tr>
<td>onRebind()</td>
<td>The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its <code>onUnbind(Intent)</code> method.</td>
</tr>
</tbody>
</table>

---

Table 3–1 Android service callback methods

<table>
<thead>
<tr>
<th>Callback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onCreate()</td>
<td>The system calls this method when the service is first created using onStartCommand() or onBind(). This call is required to perform one-time set-up.</td>
</tr>
<tr>
<td>onDestroy()</td>
<td>The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.</td>
</tr>
</tbody>
</table>

Service created in this task acquires location data and data relative to vibrations, in light of this further will detail how Android treats these issues.

In point of view of Accelerometer sensor, Android reports acceleration in m/s². Earth’s gravity is 9.8 m/s² or 1 g (gee, a unit of gravity) downwards.

However, when at rest the sensor reports its z-value to be +9.8 m/s², because it reports positive values for downwards accelerations. For all three accelerations, the convention is:

- values[0]: Minus gx on the x-axis
- values[1]: Minus gy on the y-axis
- values[2]: Minus gz on the z-axis

where gx, gy, and gz are the three components of the measured acceleration vector.

![Android coordinate systems](source Android API Guide) (3) (4)

Because we work with sensors in an Android service, first of all we implemented the SensorEventListener interface. This interface is used for receiving notifications from the SensorManager when sensor values have changed (see section 3.1).

In the process of sensor data acquisition, we’ve overridden a couple of methods – onSensorChanged and onAccuracyChanged (onAccuracyChanged is called when the accuracy of a sensor has changed and onSensorChanged is called when a sensor’s values have changed). It is shown in the following figure workflow for collecting data from the accelerometer.
From the point of view of determining a device's current location, in Android the majority of the classes that you will need when working with location data are located in the `android.location` package (see Figure 3-14).

This package consists of the following five components:

- **Classes:**
  1. LocationManager
  2. LocationProvider
  3. Location
  4. Criteria
• Interfaces:
  5. LocationListener

The class `LocationManager` provides information about the current state of the location system such as available location providers, enabled location providers, and GPS status information. The `LocationManager` can also provide the last known (cached) location of the device.

The class `LocationProvider` is an abstraction for the different sources of location information in Android. Android provides different sources of location data that have drastically different characteristics. Though each provider generates location data differently, they all communicate with an app the same way and provide similar data to an app in the same manner.

The `Location` class is what encapsulates the actual location data provided to an app from a location provider. It contains the quantifiable data such as latitude, longitude, and altitude.

An app can use the `Criteria` class to query the `LocationManager` for location providers that contain certain characteristics. The attributes on the `Criteria` class that can be used to select a location provider are the following: accuracy, altitudeRequired, bearingRequired, bearingAccuracy, costAllowed, horizontalAccuracy, powerRequirement, speedRequired, speedAccuracy, verticalAccuracy.

The `LocationListener` interface contains a group of callback methods that are called in reaction to changes in a device’s current location or changes in location service state. The `LocationManager` enables an app to register/unregister a location listener implementation that can be used to process the changes in state.

We created the service that is started from a client application (Android Activity, see section 3.1), and in this respect it is a bounded service type. This service responsible for the acquisition of location data correlated with data of vibration was set taking into account the issues mentioned above. Below is a sequence of code to the service for collecting accelerometer data and also for collecting GPS data.
Figure 3-15: Fragment of source code for Accelerometer collecting data.
3.2.2 Design of Android application gateway for data transmission

This component of the mobile system is responsible for reading data collected locally and sending them to crowd-data system via Internet.

The activities consist of UI processes responsible for setting and start / stop data transitions services.
Communication mode desired to be achieved under this component must be done (Adhoc OR Automatic by Scheduler) via RESTful-service using the phone 3G/4G or WiFi connection. Today there are many communication architectures that are based on RESTful-service, and the most important are:

- Service-centric
- ContentProvider-centric
- SyncAdapter-centric

All depend on the fact that a query to a content provider is similar to the query to the remote service for which the content provider is a proxy. In all three, resource state, including whether the resource has changed since the last time it was synchronized with the remote service, is stored locally in a content provider (backed by a SQLite database).

Finally, in all three, the task of synchronizing the data held locally in the content provider with the parallel data on the remote server, is implemented in an Android service, not an activity.
According to Zigurd Mednieks at All (2014)[5], ContentProvider-Centric is very appealing because of its homogeneity. It has a fractal-like design in which the small components look very much like the larger components that they comprise (see Figure 3-18). This is the architecture used to implement our client application (Android application is client for crowd-data system).

In this approach, the activity makes RESTful calls to a content provider as if all of the data were local. Hidden behind the content provider is a service helper that forwards requests to a service. The service is responsible, as it was in the previous approach, for synchronizing local resources with their counterparts on the remote server and doing so on a thread other than the UI (User Interface) thread. It uses a REST method to communicate with the remote server (crowd-data server). When the remote server returns a response, a processor component updates the content provider, which notifies content observers as necessary.

Since the service we created is started through a client application (Android Activity, see section 3.1), it is a bounded service type. Below is a sequence of code to our RESTful service client for communication between Android system and Crowd-Data system.
3.2.3 Design of Android application for local reporting and interrogation data

This component of the mobile system is responsible for reading data collected locally and display information in graphic format.

The activities consist of UI processes responsible for loading data from the database and displaying them in a graph.

We kept a very simple and easy to use layout (see Figure 3-20), which consists of the following elements:

- a button to start reading
- a button to stop reading and show graph
- a button to upload data from local database
- a placeholder as the chart container
3.3 Design of crowd-data system

The crowd-data system is responsible for data collection from mobile terminals via the communication module described above, and from physical point of view it is an application that contain the following functional components: data communication, processing and storing data received, user interface functions through dedicated graphical controls (database query, display information in analysis format, commands and queries toward the local device, etc.).

The reference architecture of the crowd-data system represents a view of all types of applications that have to be developed by structuring system functional areas into layers.
The reference system architecture (see Figure 3-21) demonstrates how the types of applications interact with the users, external systems (mobile systems), data sources, and services and shows how cross-cutting concerns such as security and communication impact the system layers and have to be designed together with the all system applications.

- **Presentation Layer Components**

The components specific to “Presentation layer” are the following:

- User interface (UI) components provide for the users a way to interact with the system and a format to input data acquisition and validation;
- User process components support the synchronization and the orchestration of the user interactions being used to drive the process by using separate user process components.

- **Service Layer Components**

The components specific to “Service layer” are the following:

- Service interfaces. Web services expose a service interface to which all inbound messages are sent. The set of messages that are exchanged by the service in order to perform a specific business task constitutes a contract. Service interface is a modality to expose the business logic implemented in the service to potential consumers/users;
- Message types. When data is exchanged across the service layer, the data structures are wrapped by the message structures that support different types of operations. These message types are the “message contracts” for communication between service consumers and providers.

- **Business Layer Components**

The components specific to “Business layer” are the following:

- Application facade is used to combine multiple business operations into a single message-based operation. The application facade is accessed from the presentation layer by using different communication technologies;
- Business components implement the business logic of the system. The business process consists of an orchestrated workflow and the system contains components that implement business rules and perform business tasks;
- Business entity components pass data between components. The business entities used internally in the system are data structures and are implemented by using custom object-oriented classes that represent the real-world entities of the system;
- Business workflows. The business processes involve multiple steps that are performed in the correct order and orchestrated. The business workflows define and coordinate long-running, multi-step business processes and are implemented using business process management tools.

- **Data Layer Components**

The components specific to “Data layer” are the following:

- Data access logic components abstract the logic necessary to access the data stores, centralize data access functionality, and make the process easier to configure and maintain;
- Data utility components by their functions and utilities assist the data manipulation, data transformation, and data access within the layer. These are specialized libraries or custom
routines especially designed to maximize data access performance and reduce the
development needs of the logic components and the service agent parts of the layer;
- Service agents isolate the system from the idiosyncrasies of calling diverse services and
  provide additional services such as basic mapping between the format of the data exposed
  by the service and the format that the system requires.

From physical point of view, the types of applications of system are the following:

- Web Applications designed to run primarily on the server:
  - Support multiple platforms and browsers;
  - Support only connected scenarios;
  - Uses the processing and storage resources of the server.

- Service Applications designed to support communication between loosely coupled
  components:
  - Support loose coupling between distributed components;
  - Call service using XML-based messages;
  - Are accessed from the local machine or remotely, depending on the transport
    protocol.

Table 3–2 underlines some benefits and considerations for the application types presented.

<table>
<thead>
<tr>
<th>Application type</th>
<th>Benefits</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Applications (web interface for crowd-data server)</td>
<td>• Has a standards based UI across multiple platforms</td>
<td>• Dependency on network connectivity (must be connected all of the time)</td>
</tr>
<tr>
<td></td>
<td>• Ease deployment and change management</td>
<td>• Difficulty in providing a rich UI</td>
</tr>
<tr>
<td>Services Applications (RESTful service &amp; common database for crowd-data server)</td>
<td>• Provide loosely coupled interactions between client and server</td>
<td>• No UI support</td>
</tr>
<tr>
<td></td>
<td>• Can be consumed by different and unrelated applications</td>
<td>• Client is dependent on network connectivity</td>
</tr>
<tr>
<td></td>
<td>• Supports interoperability</td>
<td></td>
</tr>
</tbody>
</table>

Table 3–2 Benefits and considerations at application type level

3.3.1 Design of RESTful service & common database for crowd-data server

This component of the Crowd-Data system is responsible for communication between Smartphone and data base of Crowd-Data system.

REST architecture will be useful to build client/server network applications. REST represents Representational State Transfer. Implementing REST is very simple compared to other methods like SOAP, CORBA, WSDL etc., it basically works on HTTP protocol.

A well-designed RESTful API should support most commonly used HTTP methods, each method should be used depending on the type of operation you are performing.

The most commonly HTTP methods the following:
The RESTful API created in our system considering operations like creating and reading information provided by smartphone system (see section 3.2). All operations related API calls should include API key in Authorization header field.

In the following table are the list of API calls.

<table>
<thead>
<tr>
<th>URL</th>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/login</td>
<td>POST</td>
<td>username, password</td>
<td>User login</td>
</tr>
<tr>
<td>/vibrations</td>
<td>POST</td>
<td>vibration</td>
<td>To create new vibration</td>
</tr>
<tr>
<td>/vibrations</td>
<td>GET</td>
<td>sessionid, railunitid</td>
<td>Fetching single specific session vibration</td>
</tr>
<tr>
<td>/vibrations/:id</td>
<td>GET</td>
<td>sessionid, railunitid</td>
<td>Fetching single specific session vibration</td>
</tr>
</tbody>
</table>

*Table 3-3 RESTful API calls*

The database structure of the Crowd-Data system is shown in the figure below.

![Figure 3-22 Database structure of the system](image)
3.3.2 Design of reporting web interface for crowd-data server

This component of the Crowd-Data system is responsible for user interface functions through dedicated graphical controls (database query, display information in analysis format, commands and queries toward the local device, etc.).

In the below figure is shown how to made a query and display information managed by the Crowd-Data system.

![Interface for display vibration for specific session](image)

*Figure 3-23 Interface for display vibration for specific session*
4 Strategies for creating an accessible and useful system for track and ride quality monitoring based on Low Cost Smartphone

Creating a data acquisition system, using the Smartphone devices from the Low Cost category, will be made after a careful selection of commercial variants on the market. Low Cost category presumes a low level of hardware and computing performance and even lack of implicit functions, required by our application, etc. These drawbacks must be corrected by defining and programming of additional software modules but also additional optimizations for those sequences of the code which must run very fast.

Application that is made in the T4.3 mainly includes software modules for data acquisition from the integrated sensors of the Smartphone device, software communication module for data transmission on long distance, having as destination an application as type of Control Centre. These modules of the program will reside in the device "on the field" that will work located in a rolling stock unit, as locomotive cab, for example. Another package of program modules will be independent by the above two categories and will be represented by a software application with role of local User Interface (GUI) for showing data as graphical representation, e.g. GIS Mapping representation, but also being an instrument for making requests and changing settings to the acquisitioning device.

Also, in the same frame of the T4.3 will be realized an application type Control Centre, whose function allows the user to receive data from the system as “in field functioning”, represented by the acquisitioning and communication Smartphone device, installed on the locomotive/wagon. This software application, from Control Centre, allows data requests from some database priori saved, visualization of data through an accessible format (e.g. GIS representation), sending interrogations and making settings changes in the system by the user interventions, etc.

In the next period, there will be organized functionality tests under real conditions, whenever it is necessary the validation of the new application modules. Test sessions will be organized for the program modules, mentioned above, so that the degree of interactivity between development side and the user requests will be highest.

A special aspect is the geospatial information acquired from the GPS receiver. By default, it will be used the GPS module integrated in the device Smartphone type. It has enough accuracy when the satellite visibility conditions are good - minimum 4 satellites are needed for a complete solution for location: Time, Latitude, Longitude, and Altitude.

For the integrated GPS receiver, the difficulty of receiving is taken into account when the satellite signal reduction due to reduced visibility inside the locomotive/wagon; Smartphone device will have priority for placing within locomotives, where vibration sensors can capture the best signal resulting from the interaction between wheels and rails. In this case, the GPS signal quality gets secondary importance and may become too weak to correct positioning. This issue will be resolved by including of an external GPS receiver into system, this receiver will be located outside the locomotive/wagon and will send the precise geolocation data to the acquisitioning device.
An external GPS receiver solution shows several variants, each with its advantages and disadvantages.

Two variants are taken into consideration, these will be analysed and tested for finding the optimal solution. The variants provide using of an external GPS receiver to transmit data wirelessly or using of a GPS receiver with the wired transmitted data.

Wireless version relieves the main Smartphone device by an additional consumption – the GPS module will have an own accumulator for consumption, but will grow into a significant percentage the system overall costs.

External GPS version with wire transmission of data to the Smartphone device, keeps costs at an acceptable level of what means low-cost, but will add GPS consumption to the Smartphone battery. As approximate value, can be considered that external receiver has an equivalent power consuming as the internal GPS receiver, in this case disabled.

Experiments to be achieved in the coming period will consider to establish optimal locations for the acquisitioning device as Smartphone type, and for the external GPS receiver.

If case of external GPS receiver, an important aspect is the electrified lines and locomotives; we have to find solutions for reducing the effect of perturbation type EMI (Electromagnetic Interference) and ESD (Electrostatic Discharge) against GPS receiver electronics and its communication interface. Also, it will be an essential aspect, the hazard reduction for placement in the top of the locomotive, which is the best solution for good satellites signals receiving.

Uniformity of application in relation to different usability situations. The designed application has to work and prove their usefulness in different situations and for different categories of railway networks. In the period of testing, will be organized sessions of experimenting to partners from Romania but also from Turkey, Slovenia, Great Britain, etc. The tests will enable uniform functionality of the system for situations such as railways used mainly for passenger transportation, freight dedicated railway lines, non-electrified lines, electrified railway lines with DC current as 3000 kVcc or alternating current as 25kVac, railroads with different types of sleepers: wood, concrete, etc.

A situation that will be common in the use of the designed system, will be the absence of the 4G support communication or frequent communication interruptions, due to poor coverage. In these situations, the system will need to work in data logger mode; the acquired data will be stored in its own memory. The experiments which will be carried out, will consider this aspect and will help dimensioning of the memory capacity, required for storing data in the local logger mode; these data will be downloaded to the end of the trip.
5 Conclusions

This deliverable has presented the two systems, which are going to be developed for the Task 4.3.

At rail unit level (System for low cost smartphone) different physical tiers of system consist in:

- Android application service for sensor acquisition data
- Android application gateway for data transmission (Adhoc OR Automatic by Scheduler) via RESTful-service using the phone 3G/4G or WiFi connection
- Basic Android application for local reporting and interrogation data

At Control Centre (System for crowd-data server) different physical tiers of system consist in:

- Web application for crowd-data server (RESTful service & common database)
- BI application (reporting web interface) for crowd-data server

In the next period, there will be organized functionality tests under real conditions, whenever it is necessary the validation of the new application modules. Test sessions will be organized for the program modules, mentioned above, so that the degree of interactivity between development side and the user requests will be highest.
6 References


