D3.2.2 – Mobility Data Integration Models and Techniques (intermediate)

Due date: 30.09.2015

Delivery Date: 02.10.2015

Author(s): Monika Pepikj (DFKI), Giuseppe Valetto (FBK), Antti Nurminen (AALTO), Konsta Sirvio (AALTO), Tom Schilling (VMZ)

Partner(s): DFKI, FBK, AALTO, VMZ

Editor: Monika Pepikj (DFKI)

Lead Beneficiary of Deliverable: DFKI

Dissemination level: Public

Nature of the Deliverable: Report

Internal Reviewers: Stefan Schaffer (DFKI), Benjamin Dittwald (Fraunhofer)
EXECUTIVE SUMMARY

In the second year of STREETLIFE, Work Package 3 (WP3) continued the work in advancing the models and techniques for mobility data integration and management. The work of WP3 is organized in five tasks: a) data definition, b) real-time data, c) crowdsourcing, d) intermodal mobility data representation and e) data correlation and analysis. In the Deliverable D3.2.2 we elaborate on the Y2 development of WP3, giving a component-based overview for easier correlation with the rest of the STREETLIFE systems and applications.

Developing a STREETLIFE crowdsourcing initiative was one of the main focuses of WP3 in Y2. We have divided this initiative into two separate modules: crowdsensing and crowdsourcing. In this deliverable we give an overview of the concepts and implementation we have done in both parts, always referring to the integration of these modules with the rest of the STREETLIFE components and application.

In parallel with the STREETLIFE scenarios and use cases development, we have further developed and updated the STREETLIFE data model. For the parking availability scenario purposes, we have extended the existing data model with historic information on data occupancy and parking meters. In the scope of the Berlin pilot development of a route planer with cycling safety features we have extended the data model with bicycle safety related attributes and accident analysis data.

In this deliverable we also present the initial data correlation and analysis techniques we developed for setting the bases of mode detection in STREETLIFE and the continuation of the work on real-time data and interest management techniques and algorithms.

At the end we provide a short discussion on planned work for Y3.

Disclaimer: This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 608991.

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

© Copyright in this document remains vested with the STREETLIFE Partners
D3.2.2 – Mobility data integration models and techniques (intermediate)

Table of Contents

EXECUTIVE SUMMARY .................................................................................................................. 2
ABBREVIATIONS ............................................................................................................................. 5
PARTNER .......................................................................................................................................... 7
LIST OF FIGURES ............................................................................................................................ 8
LIST OF TABLES ............................................................................................................................... 8
CODE LISTINGS .............................................................................................................................. 9
1. INTRODUCTION ......................................................................................................................... 10
2. CROWDSENSING ......................................................................................................................... 11
   2.1. USER TRACKING ...................................................................................................................... 12
   2.2. SHARED ECONOMY AND CROWDSENSING ......................................................................... 14
      2.2.1. Shared economy transport concepts overview ............................................................. 14
      2.2.2. Shared economy potential for STREETLIFE ............................................................... 16
      2.2.3. Bicycle fleet crowd-sensing use case ............................................................................ 18
3. CROWDSOURCING ....................................................................................................................... 21
   3.1. STREETLIFE CROWDSOURCING INITIATIVE - CONCEPT .................................................. 21
   3.2. DESIGN AND IMPLEMENTATION ....................................................................................... 22
      3.2.1. Component design ......................................................................................................... 22
      3.2.2. Component Implementation ........................................................................................... 25
   3.3. STREETLIFE APPLICATIONS THAT APPLY THE APPROACH ......................................... 30
      3.3.1. Using the crowdsourcing facility in Rovereto ............................................................... 30
      3.3.2. Using the crowdsourcing facility in Berlin .................................................................... 32
4. PARKING AVAILABILITY – DATA MODEL EXTENSION ............................................................... 34
5. REAL-TIME DATA MANAGEMENT ............................................................................................. 37
   5.1. AALTO’S NETWORKING IMPLEMENTATION ........................................................................ 40
   5.2. INTEREST MANAGEMENT .................................................................................................... 42
6. MODE DETECTION ....................................................................................................................... 45
   6.1. INTRODUCTION .................................................................................................................... 45
   6.2. MODE DETECTION APPROACHES ...................................................................................... 46
   6.3. INITIAL REQUIREMENTS ...................................................................................................... 47
   6.4. SELECTED STRATEGY FOR MODE DETECTION .................................................................... 47
6.5. INITIAL RESULTS ................................................................................................................................. 48
6.6. FUTURE IMPROVEMENTS .......................................................................................................................... 49

7. BICYCLE SAFE ROUTES ................................................................................................................................... 50

8. CONCLUSION AND FUTURE WORK ........................................................................................................... 55

APPENDIX A: LITERATURE ............................................................................................................................... 56
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Confidential, only for members of the Consortium (including the Commission Services)</td>
</tr>
<tr>
<td>ARPA</td>
<td>Advanced Research Projects Agency</td>
</tr>
<tr>
<td>BER</td>
<td>STREETLIFE Berlin-Pilot</td>
</tr>
<tr>
<td>BCU</td>
<td>Bicycle Communication Unit</td>
</tr>
<tr>
<td>CCT</td>
<td>Collaborative consumption transport</td>
</tr>
<tr>
<td>D</td>
<td>Deliverable</td>
</tr>
<tr>
<td>DIS</td>
<td>Distributed Interactive Simulation</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>FP7</td>
<td>Seventh Framework Programme</td>
</tr>
<tr>
<td>FLOSS</td>
<td>Free/Libre Open Source Software</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>IPR</td>
<td>Intellectual Property Rights</td>
</tr>
<tr>
<td>MGT</td>
<td>Management</td>
</tr>
<tr>
<td>MS</td>
<td>Milestone</td>
</tr>
<tr>
<td>NVE</td>
<td>Networked Virtual Environments</td>
</tr>
<tr>
<td>OS</td>
<td>Open Source</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software</td>
</tr>
<tr>
<td>O</td>
<td>Other</td>
</tr>
<tr>
<td>P</td>
<td>Prototype</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Units</td>
</tr>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PM</td>
<td>Person Month</td>
</tr>
<tr>
<td>R</td>
<td>Report</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>ROV</td>
<td>STREETLIFE Rovereto-Pilot</td>
</tr>
<tr>
<td>RTD</td>
<td>Research and Development</td>
</tr>
<tr>
<td>TRE</td>
<td>STREETLIFE Tampere-Pilot</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>Y1</td>
<td>Year 1</td>
</tr>
<tr>
<td>Y2</td>
<td>Year 2</td>
</tr>
<tr>
<td>Y3</td>
<td>Year 3</td>
</tr>
</tbody>
</table>
PARTNER

Fraunhofer  
FBK  
SIEMENS  
DFKI  
AALTO  
DLR  
CAIRE  
Rovereto  
TSB  
Tampere  
Logica  
VMZ

Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.
Fondazione Bruno Kessler
Siemens AG
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Aalto University
Deutsches Zentrum für Luft- und Raumfahrt
Cooperativa Architetti e Ingegneri - Urbanistica
Comune di Rovereto
Berlin Partner for Business and Technology
City of Tampere
CGI Suomi Oy
VMZ Berlin Betreibergesellschaft mbH
LIST OF FIGURES

Figure 1: Crowdsourcing classification and use in STREETLIFE ........................................... 11
Figure 2: User tracking application .......................................................................................... 12
Figure 3: Standalone crowdsensing data App display ............................................................... 13
Figure 4: Shared use of vehicles ............................................................................................... 15
Figure 5: Collaborative Consumption Transport providers in the STREETLIFE context ..... 17
Figure 6: Fleet architecture ...................................................................................................... 19
Figure 7: Bicycle communication unit ...................................................................................... 19
Figure 8: Bicycle crowdsensing data model ............................................................................. 20
Figure 9: STREETLIFE crowdsourcing component ................................................................. 30
Figure 10: Data structure for historical parking occupation information............................... 36
Figure 11: Defining a two-dimensional interest expression. All street segments within a rectangular viewing area are subscribed for updates. ......................................................... 43
Figure 12: Topological interest expression by defining a route with a 1-hop neighborhood .. 44
Figure 13: Decision space for mode detection ......................................................................... 46
Figure 14: Mode detection with Google Play services and post-processing ......................... 48
Figure 15: Route tested for mode detection ............................................................................. 49
Figure 16: Accidents with cyclist at links for the year 2013 .................................................... 52
Figure 17: Accidents with cyclist at intersection for the year 2013 .......................................... 53

LIST OF TABLES

Table 1: Filtering logic - keyword definition ........................................................................... 23
Table 2: Timing functions ......................................................................................................... 23
Table 3: Current activity and timing function correlation ......................................................... 29
Table 4: Architecture modifications to DIS for supporting Live Entities ............................... 40
Table 5: Possible sensors for mode detection ................................................................. 45
Table 6: Detected and actual modes ................................................................................. 49
Table 7: Accidents with cyclist at intersection for the year 2013 ..................................... 53
Table 8: Cycling preferences ............................................................................................. 54

CODE LISTINGS

Code Listing 1: Filter file example .................................................................................. 24
Code Listing 2: Crowdsourcing service response - example ........................................... 26
Code Listing 3: Rovereto itinerary excerpt ....................................................................... 27
Code Listing 4: Sample of the data structure for crowdsourced parking availability information used in STREETLIFE in Y1 ................................................................. 35
Code Listing 5: A lightweight entity update PDU definition ............................................ 41
Code Listing 6: Entity state dead reckoning parameter .................................................... 41
1. INTRODUCTION

The purpose of STREETLIFE Deliverable D3.2.2 is to report on work and advancements during Y2 of the project, related specifically to how mobility data is handled and integrated in the STREETLIFE software.

As discussed in the previous WP3 Deliverables, namely D3.1 [1] and D3.2.1 [2], the mobility-related data used within STREETLIFE and across different municipalities is very heterogeneous. Analogously, the techniques that we use and discuss hereby for generating, modelling and exchanging mobility data are also diverse. In this Deliverable, we focus on two specific types of data, namely:

- Integration and utilization of user-generated data in STREETLIFE, and specifically crowdsensed and crowdsourced sources of mobility data. We also elaborate on the way we analyse such data to extract further knowledge, for example how terminal-generated tracking data can be used for travel mode detection;
- Integration and utilization of data with real-time requirements.

Integrating and exploiting these two types of data is an explicit focus of STREETLIFE since its inception. User-generated data (either crowdsensed or crowdsourced) has the potential to provide a sustainable urban mobility information system with numerous data sources that supplement in reach, granularity and precision the more traditional mobility data sources that can be integrated from the city and third-party services. Data with real-time requirements caters to the increasing needs of citizens as well as mobility operators and managers for up-to-date “here and now” snapshots and updates of the mobility situation. Both types of data thus are a key component of additional and more innovative services and applications, like the ones we pursue in STREETLIFE.

In D3.2.2 we presents some notable examples, which show how the STREETLIFE data model can combine and leverage various data sources, including, prominently, user-generated and real-time data to provide views on mobility data that were not previously available, and which augment the sustainable mobility services that STREETLIFE can make available to citizens as well as municipalities.

We provide two such examples. The first example discusses the parking availability use case, which has interesting implications on sustainable mobility management capabilities for the city administration, on recommendations and impact awareness for the individual citizens, as well, as technically speaking – for multi-data source integration within the STREETLIFE mobility information system. The second example discusses support for the bicycle safe routes use case, which can significantly impact behavioural change, and bring augmented awareness to citizens using STREETLIFE technologies. That is a clear showcase of how STREETLIFE adds value to individual data sources by integrating them in ways that are important for smarter mobility and can effectively facilitate citizens in choosing their mobility options.

This report, therefore, aims at showing how STREETLIFE, by being able to incorporate and handle more and more types of data from diverse sources, and by combining them in various
ways, is incrementally achieving enough reach and versatility to provide an increasingly wider range of information and functionality. We also show that the data model can offer a large number of innovative services, as required to support contemporary smart and sustainable mobility requirements.

2. CROWDSENSING

User-generated information is a very promising source of data for the purposes of STREETLIFE, since it has a potential of enhancing the quality and timeliness of ICT solutions that address and support urban sustainable mobility, all the while improving the level of awareness and participation of involved stakeholders (first and foremost regular citizens, but also different kinds of operators and officers involved in the city mobility system).

In this Section we discuss one type of such user-generated mobility information, i.e. crowdsensing data and its applications in STREETLIFE. Crowdsensing is (according to the classification we offered in D3.2.1 [2], and which we report below in Figure 1 for convenience) a form of user-generated data that is passive and dynamic.

As elaborated in D3.2.1 [2], in STREETLIFE we use crowdsensing for two purposes: user-centred for personalization purposes (Section 2.1) and crowd-centred for gaining more general mobility information (Section 2.2). In STREETLIFE we put an accent on the bicycle-related crowdsensing and its potential (Section 2.2.3).

Figure 1: Crowdsourcing classification and use in STREETLIFE

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td></td>
<td>Travellers tracking (BER)</td>
</tr>
<tr>
<td>Active</td>
<td>[Open Street Map (ROV, TRE)]</td>
<td>Users’ notifications (BER, ROV, TRE) Traffic aides’ notifications (ROV)</td>
</tr>
</tbody>
</table>
2.1. User tracking

Contextual user state tracking has multiple potential applications, ranging from simple features within context-aware software to live social media as real-time extension to micro-blogging. Social-driven location sharing has already become a pervasive phenomenon in contrast to purpose-driven sharing [3]. Along with the emergence of smart phones a decade ago, simple sharing of automatically determined availability data improved the enjoyment of using a phone’s contact list, expanding its original functionality [4].

The first year development and results in STREETLIFE have shown that many components would improve their functionality if user tracking were available. Therefore we extended the STREETLIFE data model with user tracking data.

Figure 2 shows the current set of components that use the user tracking data in STREETLIFE. We have divided these component into two main groups: components that use the current sensor data for online analysis and components that use collected user tracking data for offline analysis.

In a pragmatic approach for sensing real-time user data, we are utilizing capabilities of modern Android tablets and phones and external devices.

The basic sensor required for tracking is the user position (GPS). The number of other sensors that are going to be tracked is flexible and depends on the use case and the components that are using the tracking data.

For mode detection purposes (Section 6) we have created an Android Activity Recognition standalone App for collecting and transmitting this data in real-time over wireless networks.
Figure 3 presents the simple status view of the App. The App runs on the background, transmitting state updates as needed. These states are *published* by the App, and are propagated to any implicitly or explicitly expressed *subscriber*.

The tracking functionality is also part of the Mixed Reality App [5].

Our current user state for this App consist of the following fields:

- Position (internal or external GPS)
- Orientation (internal IMU)
- Heart rate (external heart rate device)
- Illumination (internal sensor)
- Ambient noise (microphone)
- Ambient and device temperature (internal sensors)
- Device tilting (Google Play services)
- Approximate modality as a set of probabilities (Google Play services)
- User-selectable modality (in addition to Google Play services estimate; optional)

![ActivityRecognitionLogger](image)

*Figure 3: Standalone crowdsensing data App display*

Currently, the modality estimation data is transmitted directly as measured by Google Play Services. In future, we will add activity recognition logic to utilize external data sources and short-term historical data to improve the accuracy. This result will then be included as an additional field to the original data. All data submitted from the App is stored in a database for later retrieval and analysis. We will use such stored data for later simulations.
Our crowdsensed data will also be analysed and provided for further aggregation or offline uses by external systems via standardized interfaces such as Datex II. Real time requirements for such data are not critical, and propagation of it can therefore be handled with more traditional components, i.e. as web services.

In the case of the Mixed Reality App the focus in crowdsensing data gathering is in the engineering challenges of real-time data distribution and management in large-scale systems. The main use case for this is the direct visualization in the advanced graphical interfaces of WP5 (D5.2.2 [5]). For example, in the augmented reality (AR) part of the Mixed Reality App, we present labels of bus line numbers on top of tracked buses; the exact position of each bus should be transmitted to the App instantly to avoid disambiguation.

In addition to the engineering challenge, real-time user tracking data aims at being as rich as possible, facilitating not just visualization and analysis, but further App features. We express user-tracking data as states, of which position is only one. Given available programming interfaces, we extract contextual and situational data, such as heart rate and estimated modality (activity). For a traffic case, the contextual data can be used to improve the robustness of mode detection.

Current user state and real time analysis are also the basis of the crowdsourcing (Section 3) and route deviation recognition (D5.2.2 [5]) component. Here user-tracking data is used to find correlation between user’s planned and actual route and determine the context.

### 2.2. Shared economy and crowdsensing

#### 2.2.1. Shared economy transport concepts overview

The modern version of shared economy concept is based on the publication in “Community Structure and Collaborative Consumption: A routine activity approach” [6]. The main idea of shared economy is collaborative consumption, where ownership of a commodity is separated from right to use a commodity. The greatest benefits from collaborative consumption are achieved when the number of users is high, utilization rate is low and the price of the commodity is high, which translates mainly to passenger cars in the transport sector.

Shared use of passenger cars (or other vehicles) can be categorized according to Figure 4.
In case of carsharing the idea is that the user is driver and he can decide the route and number of passengers. Ridesharing, however, does not necessarily allow passengers to decide exact start and end locations. If one of the passengers is a dedicated driver the start and end destinations are usually selectable by the passenger(s).

Besides carsharing, bicycle sharing systems are growing in popularity in many big cities [7] and are becoming an integrated part of multimodal transportation systems.

Here we give an overview of the different types of systems for shared use of vehicles, addressing the bicycle-sharing concept when applicable.

**Traditional vehicle sharing clubs:** Vehicle sharing is a form of car rental or bike rental, where people can rent vehicles on pay as you go basis or on a monthly or yearly membership fee. The vehicles are spread around to designated areas, from where members of the sharing scheme can pick them up on self-service either on go or after a prebooking.

**Peer-to-peer vehicle rental:** Peer-to-peer (p2p) car rental takes the carsharing into a new level. The main difference in comparison to traditional carsharing is that there is no company owned rental fleet. It is ordinary private people, who are renting out and thus sharing their vehicles with the public. The basic idea behind peer-to-peer car rental service is that the private car owners make their vehicles available for rental to their neighbours and general public alike, when they do not need the car themselves. That will allow car owners to earn money with their vehicles and to save in costs of owning a car. While for people who do not own a car, the scheme provides an opportunity to get a car conveniently, when needed, and within the neighbourhood or even from next door.
The latest technology development in the bicycle industry: smart locks (BitLock [8]) and bike computers allow users to unlock the bicycles with their smartphones and to give authority to other smartphone users to do so. With this the peer-to-peer bicycle sharing is being developed.

**Corporate vehicle sharing:** In corporate carsharing the cars are shared for the employees during the working hours and they can be rented out after work and during weekends to the employees or outsiders.

Corporate carsharing has some distinguishing features compared to the traditional carsharing clubs:

1. The service can be retrofitted to existing cars, which means that the car makes and models are not uniform
2. Cars do not necessarily have logos on the surface and therefore identification of them may be difficult only by registration plate, car make and model and GPS coordinates
3. Corporate carsharing cars are predominantly reserved for work trips during the weekdays, while they can be available for registered users during evening and weekends
4. Reservation, door unlock/lock is handled with a mobile phone

Similar applies to corporate bike sharing. In corporate bike sharing, the companies provide bicycles that can be shared among employees. Google is a great example for corporate bike sharing.

*Carpooling:* Carpooling or ride sharing is the sharing of car journeys so that more than one person travels a car increasing thus capacity utilisation rate and decreasing the price and emissions per passenger. Mobile devices with GPS sensors and continuous data connection have enabled dynamic ride sharing for ad-hoc travelling needs.

**Ride sourcing:** Ride sourcing is a concept, where the drivers do not share the destination with the passengers. The rides are outsourced to professional drivers with the help of mobile applications. Uber [9] and Lyft [10] are examples of ride sourcing services.

### 2.2.2. Shared economy potential for STREETLIFE

In STREETLIFE, we distinguish two different direction of collaborative consumption transport (CCT) integration: a) direct connection with the CCT providers and integration of services in the larger STREETLIFE multimodal router and b) crowdsensed data collection from CCT users and transformation of this data into models and services.

The STREETLIFE systems in ROV and BER are already integrating the local bicycle sharing system into the multimodal trip plannerners. The information is received from a CCT provider’s API to the server according to Figure 5. When a user is looking for a multimodal route CCT modes will appear if they are enabled and meet the personalisation criteria.

The main information required from a CCT mode for integration purposes is the following:

1. Vehicle type and identification information
2. Vehicle location information  
3. Vehicle availability time information  
4. Price information

**Figure 5: Collaborative Consumption Transport providers in the STREETLIFE context**

As other possible use cases for collaborative consumption transport (CCT) integration in the larger framework of STREETLIFE transport concept we recognize:

**Location and visualisation of car:** Location information of the selected car is visualised on a STREETLIFE application in a convenient way. In the mixed reality App a 3D model of the car can be visualised real-time in the right location.

**Navigation to a car or ride location:** If access to the car requires walking a STREETLIFE application can navigate the user to the right location, where the selected car can be found.

**Multi-modal routing including a CCT mode:** As mentioned above, so far, the multimodal trip planners integrate local bicycle sharing systems. Carsharing can be also a part of longer multimodal transport chain. In this case the best suitable car is included in the multimodal route.

**Presentation of multimodal routes including a CCT mode:** A multimodal route can be presented in a STREETLIFE application with symbols distinguishing a CCT transport mode from
others. Visual presentation of the route can include the route of each mode and mode change locations and times on the map with distinguishable symbols for each mode.

**Notification of multi-modal route changes with a CCT mode:** In case of any multimodal route change notifications can be sent including a CCT mode.

Floating data collection or crowdsensing has been already used for obtaining travel time and speed data from vehicles and present essential source for real time traffic information. It is based on the collection of localisation data, speed, travel direction and time information from mobile phones, GPS devices or other sensors available in vehicles. Shared vehicles have even greater potential in crowdsensing because they are used by different people and have greater diversity of roads when compared to private vehicles. The traceability and data collection from shared vehicles has been used for traffic planning and efficiency in the cities. Research has been using traces from taxi drivers to develop driver’s behavioural models [11] or gain general traffic image. In STREETLIFE we explore the crowdsensing potential of bicycle sharing systems. In addition to the above mentioned use-cases where vehicle tracking is mainly used for obtaining the current location, we proposes to use floating data collected from bicycles as a source for defining cycling routes directly from cyclists’ experience.

2.2.3. Bicycle fleet crowd-sensing use case

Bicycle sharing systems present a valuable source of diverse traveling information that covers larger area of the city. Nevertheless, the product and application market for bicycles is still motivated by cycling as a sport and very little attention is paid to cycling as transportation. Urban cycling in research has been addressed from a route choice perspective [12], [13]. The attributes considered for these models are mainly road and infrastructure related.

When compared with cars or public transport, the technology support like navigation and personalized assistance for cyclists is very poor. In STREETLIFE, by collecting relevant data and developing services we are addressing the urban cycling support. We research the potential of the sensor-equipped bicycles as part of the smart mobility system in a city. Based on variety of sensors, knowledge about influencing factors and relevant cycling situations will be derived. This knowledge will be used in STREETLIFE for developing real-time services for cyclists to foster carbon-low mobility in cities and communities.

The architecture of a sensor-equipped bicycle fleet is shown in Figure 6. Its three main parts are: the bicycle communication unit (BCU), backend server and frontend services.
The BCU (Figure 7) consists of a sensor unit (Arduino based) currently hosting GPS, barometric pressure, ambient temperature, proximity, loudness and CO$_2$ -sensor and a computational unit (RaspberryPi based) responsible for data collection and communication with the rest of the system (the server and the user devices).

**Figure 6: Fleet architecture**

The server unit is the central data correlation unit. The current data model for sensor data collection is given in Figure 8. Data analysis and learning algorithms will be developed to provide data for the cycling data model. Two state brokers, one for the real-time state of the sensors and the other one for the aggregated knowledge on bicycles and routes provide access to the collected data. The circle of communication is closed by the services on the user devices. Both, the server and the BCU are providing APIs for bidirectional data exchange. The current set of categories of data that the applications can get access to is: bike state, cyclist state, traffic state, context and routing.

**Figure 7: Bicycle communication unit**

The BCU (Figure 7) consists of a sensor unit (Arduino based) currently hosting GPS, barometric pressure, ambient temperature, proximity, loudness and CO$_2$ -sensor and a computational unit (RaspberryPi based) responsible for data collection and communication with the rest of the system (the server and the user devices).
Figure 8: Bicycle crowdsensing data model

The questions we are addressing by collection and observation of crowdsensed cycling data are:

Can sensors and actuators contribute to new cycling route choice models and real-time cycling event detection?

Does introduction of aggregated sensor and crowdsourced data into navigation and route representation have added value for the cyclists?

At the Saarland University campus we did the first field test, collecting cycling data from 22 participants. We collected data from the BCU mounted on the bicycles and from the phones also mounted in front of the cyclists. At the time of writing we are working on the data analysis. We are going to aggregate the taken routes into crowd-generated routes on the campus and evaluate these routes against the routes suggested by available route planers like Open Trip Planner or Google maps. We are also going to do aggregate the other sensors and provide additional route information for the routes: average CO\textsubscript{2}-emission, average speed, average loudness etc.
3. CROWDSOURCING

In this Section we describe how STREETLIFE supports and makes use of active and dynamic user-generated data (for brevity, only “crowdsourced” in the remainder).

3.1. STREETLIFE crowdsourcing initiative - Concept

Crowdsourced data in STREETLIFE was largely limited, during the first project iteration, to information originating from specialized stakeholders that have an institutional role in the support of urban mobility. This was exemplified by the App “Conta Parcheggi”, deployed in the ROV pilot, which enables crowdsourcing by city traffic aides in order to build an overview of the parking availability situation in the city (D5.2.2[5], D5.2.1[14]). That crowdsourced data, in turn, underlies monitoring and decision-making capabilities when presented to mobility managers, and estimation of additional times and costs related to finding parking spots when presented to citizen planning a car itinerary.

That kind of “specialized” crowdsourcing was successful in Y1 of the STREETLIFE project. In fact, “Conta Parcheggi” was deployed and used with satisfaction, and finally adopted by the Municipality of Rovereto, which has led to a new productized release (see also D.5.2.2 [5]) in Y2. However, given its specialized purpose, it covers only partially the intended scope of crowdsourcing in STREETLIFE, which includes crowdsourcing as a form of active engagement of the general public as part of the smart and sustainable mobility system of a city.

In the current iteration of the project, we are therefore focusing on extending crowdsourcing capabilities to regular citizens who interact with the STREETLIFE Apps and services. In defining use cases, requirements and capabilities for that, it is important to consider crowdsourcing applications that may work across the very diverse urban contexts of European cities, exemplified in the three pilot cities of ROV, TRE and BER. For instance, crowdsourced data sources that rely upon volume and frequency, that is, statistics on many repeated and frequent samples, such as the crowdsensing of transfer speed on city roads (like in the commercial Waze App [15]), or the notification of actual passage, delays and transfer times of public transportation vehicles, may provide effective benefits only when specific spots are hit with respect to city size and penetration rate of the App in the city. This is an important lesson to consider; in fact, some of the Apps geared toward the general public that we had deployed in the STREETLIFE pilot cities (for example in TRE and ROV) already offered – originally – notification features for active crowdsourcing. However, they were primarily geared toward volume- and frequency-based crowdsourcing, hence they struggled to reach the necessary critical mass.

In general, it becomes increasingly difficult to obtain the necessary statistical relevance as the size of the city becomes smaller. There are other use cases of crowdsourced information, however, that are largely independent from data volume or frequency, and hence of more widespread applicability across diverse urban environments; for instance, participatory notifications of specific road, vehicle or infrastructure conditions and incidents may need a single piece of data, or at most a few confirmations; recommendations need multiple repeated samples for reinforcement, but their frequency is not critical; etc.
In consideration of the issues mentioned above, we have developed a facility that allows STREETLIFE to collect crowdsourced information that is *customized, contextualized and timely* from individual users of STREETLIFE Apps, and which is equally appropriate for usages that rely on volume and frequency of the crowdsourced data source, as well as usages that are independent of that. The concept is based on a “push mechanism” that presents to the App user a multiple-choice form, corresponding to a specific sustainable urban mobility question or concern of the city. The question is specific to the user’s itinerary and is presented at specific junctures during the itinerary; this timing is important because the STREETLIFE crowdsourcing facility strives to ask the question when the user is most likely to be able and willing to answer. For example, a question asking to rate a bike route should be asked when it is safe to grab the user’s attention (e.g. after the bike leg is over), whereas a question about the comfort of a public transport vehicle should be asked in the middle of the public transport leg, when the user is at ease, mostly idle, and can directly assess the conditions of the vehicle, instead of resorting to memory for her answer.

The assumption of the STREETLIFE crowdsourcing facility is that precise contextualization and timeliness with respect to the user’s experience in the course of the itinerary is a key for engagement, as well as for being able to obtain user-generated data that can be effectively leveraged as an additional rich and reliable data source within the smart and sustainable mobility system of a city.

In order to decide what questions are contextually relevant and when is the right time to push them to the user’s device, the STREETLIFE crowdsourcing facility must make use of itinerary tracking and mode detection functionality that is included in the route planning and travel assistance components of STREETLIFE (see Section 6 and [5] for details on those).

### 3.2. Design and Implementation

As described above, the main idea of the STREETLIFE crowdsourcing initiative is to ask the right questions at the right time. With initial assumption and by user monitoring and learning, we intend to learn to detect the context of a multimodal trip and decide when would be the best fitting time to ask a mobility-related question.

#### 3.2.1. Component design

Different situations and different crowdsourcing goals have different conditions under which the user should or should not be presented with a question. This concept of mobile crowdsourcing takes in consideration the sensed state of the user and her context or environment, and proactively asks for user participation. The participation can take different forms like: answering an open or closed (multiple-choice) question, uploading a photo, marking or confirming a position etc. With this approach, we believe we will avoid the problems like: losing interest to enter data after the trip, not remembering details about the trip, not caring about irrelevant questions with respect to the user’s experience etc.

The concept we have developed has three main building blocks:
Questions: Defines the trip/mobility related questions that we want to ask the users. Even though the concept itself does not restrict the question and answer form, our initial assumption (also learned from previous partners’ experience) is that the questions need to be concrete and well defined. The answering methodology should fit to the traveling engagement or the users. Examples of questions and answers we designed for different crowdsourcing situations are given in the descriptions of the STREETLIFE crowdsourcing applications (Section 3.3).

Logic or filtering conditions: For each question or group of questions we define the filtering logic. The logic defines the conditions that make a trip a candidate for posing the question to the corresponding user. To have bigger flexibility, we have defined this logic as combination of predicates upon trip legs and attributes.

Table 1: Filtering logic - keyword definition

<table>
<thead>
<tr>
<th>keyword</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>filterBlock AND/OR/NEXT filterBlock</td>
</tr>
<tr>
<td>filterBlock</td>
<td>transportation AND transportation_filters</td>
</tr>
<tr>
<td>transportation</td>
<td>bus, walk, bicycle, train, car</td>
</tr>
<tr>
<td>transportation_filters</td>
<td>preferences: fast, green, comfortable, etc.</td>
</tr>
<tr>
<td></td>
<td>general: duration</td>
</tr>
</tbody>
</table>

The filter logic can have one or two filter blocks that are connected with an operator (Table 1). Each filter block refers to a route leg. The whole logic expression refers to the whole route. This gives us flexibility to support and research the multimodal transportation. Namely in case we want to consider multimodal trips with two different modes and ask the question only in this case, then each filter block will refer to one mode and the operation will define the order how these modes should be considered (AND – both need to appear, the order does not matter, OR – at least one needs to appear, NEXT – both need to appear and the order matters).

Timing: The timing when to ask the question is defined through user position relative to (parts of) the trip. At the moment we support four relative positions described in Table 2.

Table 2: Timing functions

<table>
<thead>
<tr>
<th>atStartPosition(leg*)</th>
<th>at start position of a leg, before taking the planned modality</th>
</tr>
</thead>
<tbody>
<tr>
<td>atEndPosition(leg)</td>
<td>at end position of a leg, after leaving the current modality</td>
</tr>
</tbody>
</table>
during(leg) | during a given leg. The timing at the moment is set to be approximately in the middle of the leg.
--- | ---
atEndPosition() | at the end of the whole trip

*leg = logicBlock - defined by the transportation

An example of a configuration file for the STREETLIFE crowdsourcing facility that puts these three concepts together is shown in Error! Reference source not found.

```json
{
    "crowdsourcing_situations": [
        {
            "id": "dropOffStation",
            "filter": {
                "filterBlocks": [
                    {
                        "transportation": "bicycle",
                        "attribute": null
                    }
                ],
                "operator": null
            },
            "timing": {
                "function": "atEndPosition",
                "leg": "bicycle"
            },
            "question": {
                "q": "Please rate the drop-off station we suggested:",
                "answers": [
                    "Enough free parking places, station at a good location",
                    "Enough free parking places, station at a bad location",
                    "I took the last parking place, station at a good location",
                    "I took the last parking place, station at a bad location",
                    "No place to park my bike"
                ]
            }
        }
    ]
}
```

**Code Listing 1: Filter file example**
3.2.2. Component Implementation

We have developed a crowdsourcing component that implements the logic and supports crowdsourcing initiatives to STREETLIFE mobility Apps (and can be extended to other Apps as well).

The component has two main parts:

**Crowdsourcing service**: The crowdsourcing service is a java RESTful service that takes the configuration file (Code Listing 1) and an itinerary file (Code Listing 3) as an input and based on the filters from the configured crowdsourcing situations selects the relevant situations for the given trip. If every constraint is met, the service will respond back with a JSON object containing the selected question and the timing for asking that question to the end user (Code Listing 2).

The Service has the following functionality:

- Upload a configuration file. Configuration file is meant to be created once by the application owner or the crowdsourcing use-cases creator and to be uploaded on the server. The file can be uploaded by a POST call with the following URL http://www.dfki.de/restfull/CrowdSourcingService/home/crowdSourcing/uploadFilter

- Send a trip file and retrieve the relevant questions and timing. When a client sends a trip file, the service parses the content and creates a corresponding trip object for internal processing. Then it checks if the trip satisfies filter blocks and other constraint and returns the corresponding questions and timing to the client.

Call details:

**URL**:
http://www.dfki.de/restfull/CrowdSourcingService/home/crowdSourcing/uploadFilter/uploadTrip

**Method**: POST
**Data type**: JSON

**Response**: A list of crowdsourcing situations. It can be empty or containing one or more crowdsourcing situations that meet the filters form the configuration file (Code Listing 2).
Code Listing 2: Crowdsourcing service response - example

```json
{
    "situations": [
        {
            "tripId": "54632f3e975a91904bbdff42",
            "timming": {
                "function": "atStartLegPosition",
                "leg": "bicycle"
            },
            "question": {
                "q": "Please rate the drop-off station we suggested: ",
                "answers": [
                    "Enough free parking places, station at a good location",
                    "Enough free parking places, station at a bad location",
                    "I took the last parking place, station at a good location",
                    "I took the last parking place, station at a bad location",
                    "No place to park my bike"
                ]
            }
        }
    ]
}
```

```json
{
    "alertAccidentList": [],
    "alertDelayList": [],
    "alertParkingList": [
        {
            "_id": "Mart - Rovereto@BIKE_SHARING_TOBIKE_ROVERETO",
            "creatorId": "",
            "creatorType": "SERVICE",
            "description": "Bike Station Mart - Rovereto: available bikes = 1, available parkings = 4",
            "from": 0,
            "noOfvehicles": 1,
            "place": {
                "_id": "Mart - Rovereto",
                "agencyId": "BIKE_SHARING_TOBIKE_ROVERETO"
            },
            "placesAvailable": 4,
            "to": 0,
            "type": "PARKING"
        }
    ]
}
```
Code Listing 3: Rovereto itinerary excerpt
Crowdsourcing App Library: The second part of the crowdsourcing component is the crowdsourcing library. The main functionality of this library is to implement the timing on the client side. Once having questions and timing data, the library is used in combination with the user tracking component and mode detection component to determine if the user has entered a specific leg at a specific position of the leg, and to decide on the best time to pop up the questions.

Requirements from the application that will use this library are:

- Access to Internet: to connect to Crowdsourcing Service
- User current location: to detect whether a user has entered a given leg at a specific position
- Activity Recognition: to detect user’s activity and mode (On vehicle, On foot, Running, Still, etc.)

The component supports at the moment the timing functions given in Table 2: atStartPosition(leg); atEndPosition(leg); during(leg) atEndPosition()

As already elaborated in Section 2.1, we are using the Google activity detection API in combination with the current position and the leg geometry to define the trip context for the user. Given the possible activities detected from the Google API we apply the logic for the functions as shown in Table 3.

Example (refers to Table 3): In case the timing is atEndPosition(bicycle) then we monitor first that the user was on the bicycle (ON_BICYCLE activity detected and user position is on the bicycle leg) and that when the user approached the end of the leg the activity changed to one of the following: IN_VEHICLE, ON_FOOT, STILL or WALKING. We also consider UNKNOWN as a possibility in this first phase.
<table>
<thead>
<tr>
<th></th>
<th>Bicycle leg</th>
<th>Public transport (bus, train) leg</th>
<th>Own car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>atStartPos(leg)</td>
<td>During atEndPos(leg)</td>
<td>atStartPos(leg)</td>
</tr>
<tr>
<td>IN_VEHICLE</td>
<td>X</td>
<td>X</td>
<td>Required</td>
</tr>
<tr>
<td>ON_BICYCLE</td>
<td>X</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td>ON FOOT</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RUNNING</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STILL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TILTING</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WALKING</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 9 depicts the two main parts of the STREETLIFE crowdsourcing component and their relation with each other and with other components.
3.3. STREETLIFE applications that apply the approach

3.3.1. Using the crowdsourcing facility in Rovereto

The size of the city of Rovereto is not sufficient to sustain a volume- or frequency-based crowdsourcing initiative. In fact, past experience, with this kind of crowdsourcing data, although already supported by the “ViaggiaRovereto” mobile App, were not successful, even if its market penetration rate among citizens and visitors of ROV that employ mobile Apps to support their mobility needs is quite significant (as discussed for example in D6.2.1 [16]). The same was true the ROV Bike Sharing App, which has also been used in STREETLIFE. The bottom line of past experiments is that not enough users take the initiative on their own of exercising the crowdsourcing features of the Apps, and thus the user community cannot produce enough crowdsourced information for volume- or frequency-based application of that information.

The most promising use case for the STREETLIFE crowdsourcing facility in the context of the ROV pilot is therefore about proactively asking citizens to issue notifications on specific road, vehicle or infrastructure conditions and incidents they become aware of, as they use them or during their itineraries. In these cases, even a single notification (or a few mutually confirming notifications) represents useful user-generated data that can be incorporated in the smart mobility system of the city. Because of that, the crowdsourced data can become useful even in case of low yield from the end users.

A specific example we are planning to implement in the second iteration ROV pilot, to validate the STREETLIFE crowdsourcing facility, is centred upon the recently introduced ROV bike sharing service. The crowdsourcing facility will be used to ask users questions about the bike sharing service, at either the beginning or the end of a bike sharing leg in their chosen itinerary. Some questions aim at retrieving information about the operational conditions of vehicles users...
see at both end points of the leg (i.e. when they reach the start and end bike sharing stations). Such a question may take the following form:

“Are there any bikes that need maintenance at the sharing station where you are?”

The question takes a YES or NO answer, with the possibility to add a vehicle number and a comment in case the answer is YES.

Analogously, the question may regard simply the bike being ridden by the user, in which case a question in the following form should be asked once the user has reached the end point of the bike sharing leg:

“How do you rate the conditions of your bike sharing vehicle?”

Answers to these questions can be on a scale, for example:

- Perfect
- Satisfactory
- Functional
- Unsatisfactory
- Damaged

with the possibility to add a comment in case the rating chosen is “Damaged”.

This way, a user is encouraged to notify in a quick and compact way the municipality and the bike sharing service provider of different kinds of bike issues, such as for instance:

1. “flat tire on bike #106 at the station in Piazza Rosmini
2. “broken tail light on my bike #103, now parked at the MART station”

A similar question can be asked about the operational conditions of the start or end station itself, in a form like the following:

“Did you observe there any problems with bike slots at the sharing station where you are picking up/releasing your bike sharing vehicle?”

The question takes a YES or NO answer, with the possibility to add a slot ID number and a comment in case the answer is YES. This question can lead to the signalling of issues such as, for instance:

3. “Bike slot #2 at Piazza Rosmini station fails to charge electric bikes”
4. “Bike slot #1 at Via Paoli station fails to lock bikes”
The information provided through these use cases of the STREETLIFE crowdsourcing facility can be promptly integrated with – and leveraged by – other STREETLIFE components, such as the ROV Bike Sharing App, the ROV journey planner, and the city mobility dashboard. For example, a user of the Bike Sharing App can avoid trying to pick up a bike from the Piazza Rosmini station, if the only bike present is #106 and it has been signalled as disabled (see notification example 1. above); similarly, the journey planner can select a different return bike station, in place of Via Paoli, for its bike sharing itinerary recommendations, in case the only free slot for returning a bike at Via Paoli is slot #1, which has also been signalled as malfunctioning (see notification example 4. above).

Another use case of the STREETLIFE crowdsourcing facility is simply to ask a question to verify that the number of available bikes (or free bike slots) the user sees at the bike station she is at match the number recorded in the information system of the bike sharing service, which is also used by STREETLIFE Apps and services. Although those numbers are accurate most of the time, they may be off sometimes, for example due to sensor problems, or other issues that put the information system out of sync with the reality on the ground. Such imprecisions may lead to lessened user experience and decreased satisfaction with the ROV bike sharing service, which sometimes register as citizens’ complaints. The usage of this crowdsourced information by the STREETLIFE sustainable mobility system can be used for verification and augmentation of the data from the bike sharing operator. The way it works is analogous to the examples above, that is, provide end users as well as service operators and city mobility managers with augmented and more accurate or more timely data about some part of the city sustainable mobility system.

Other different usages of the STREETLIFE crowdsourcing facility are also possible; for example, it can be used immediately after the end of a bike sharing leg to ask the user about the level of comfort and satisfaction (on a scale) which she had in her experience with the ROV bike sharing service.

3.3.2. Using the crowdsourcing facility in Berlin

In Berlin the STREETLIFE crowdsourcing initiative will be used for two purposes: to get a subjective opinion on the safety of the proposed bicycle trips and to assess the bicycle parking possibilities in the city.

1. Single mode trips: Bicycles - addresses the cyclist safety

At the end of a bicycle trip we ask the user to give a feedback on the safety of the route

Question:

1. Rate you cycling trip:
   - Very dangerous
   - Partially not safe
   - Few challenging(stressful) points
   - Safe but difficult
• Safe and pleasant

If the user answers with any of the first three, he will be asked to mark the dangerous points on the map afterwards.

2. Multimodal trip: bike + other Modality
Issue: available safe parking for bicycles

At the end of a multimodal trip, the user is asked to rate the parking facilities for his/her bike.

Question:

Combination of two questions

1. How did you park your bike?
   • bike rack
   • bike lock
   • street parking
   • no parking was available
   • other

2. How would you rate the safety of your bike at the parking place
   • very safe
   • safe
   • moderate
   • somehow unsafe
   • unsafe

3.3.3. Using the crowdsourcing facility in Tampere
The TRE mixed reality app incorporates a crowdsourcing part to provide user feedback on a bus trip – with the related bus line and bus station:

1. Bus Line
   • pleasantness
   • easiness of use
   • difficulty

2. Bus station
   • pleasantness
• easiness of use
• difficulty

Feedback responses are combined with a timestamp and location (latitude and longitude), and transmitted to a server hosting a database. Supporting STREETLIFE’s strategic approach towards a smart city reference architecture (D2.2.2 [17], Section 8), where open data and standardisation are viewed as enabling factors, this data is provided freely in Datex II format via a web service for further aggregation and analysis.

4. PARKING AVAILABILITY – DATA MODEL EXTENSION

We have extended the urban parking availability data model that was already in place in Y1 of the STREETLIFE project, and supported, for example, by STREETLIFE elements like the crowdsourced information source from the “ContaParcheggi App” of ROV, or TRE parking availability. This as part of the integration task is elaborated in D2.2.2 [17] and D4.2.2 [18]

The significance of the urban parking availability scenario in STREETLIFE is that it provides a good example of how a multiplicity of data sources and software components come together to make the urban mobility system smarter and more sustainable. In this case, accurate and timely information about occupancy rates of parking spots in heavy-traffic areas of the city (including both parking lots occupancy and on-street parking occupancy) is offered as a service, and leveraged by the STREETLIFE system for multiple purposes: a) inform city mobility managers, parking operators and traffic aides of specific congested macro-areas (e.g. a city neighbourhood) and micro-areas (e.g. a street, a square or a stretch of road) in the city, via reporting to the mobility dashboard, or other connected information systems and services; b) adjust the journey planner, so that it takes into account such parking congestions in its itinerary recommendations to citizen and visitors; and c) transfer the information to the route planning and travel assistance components of STREETLIFE, which can leverage it to help the user in her itinerary selection, for example by showing an estimate of the (otherwise hidden) extra time and/or cost she can incur when deciding to travel by car, and park in those congested areas.

Coming to the specifics of how the data model supports the concepts above, in D3.2.1 [2] we described the data model features that were present in Y1 for this particular data. To recap, the data record on parking availability that we have been using represents a single time-stamped observation of a micro-area within a macro-area or, alternatively, of an entire parking lot.
Code Listing 4: Sample of the data structure for crowdsourced parking availability information used in STREETLIFE in Y1

That data structure (reported for the readers’ convenience in Code Listing 4) represents the basis for an extension that we have carried out in Y2 of the project. Specifically, we have extended the data model for parking in two ways:

- We have added data structures to hold historic information about parking occupancy rates. This information is built on top of the accumulated observations records, like that of Code Listing 4, and enables a multi-faceted look at average occupancy rates;
- We have added data structures to hold historic information about parking revenue information, for individual parking meters located in the city, as well as parking lots. We assume that time-stamped individual observations about parking revenue are available by connecting to ICT services of the parking operators and the city parking authority. A snapshot of the history of paid parking slots in a city (especially in the premium and often-congested city centre areas), when compared with the history of the parking occupation, is a very useful basis for decision- and policy-making relate to the urban traffic situation: for example, it could open the way to timely and even fully
dynamic decisions on changing the parking rates to de-incentivize motorists to travel to, and park at, congested locations at critical times.

Both of these historic data structures are organized as a hierarchy, whose nodes provide different ways to aggregate the individual observations logged by the city information sources integrated in STREELIFE. Figure 10 shows the hierarchy for the parking occupation rates (the hierarchy for historic parking revenues fully mirrors this one, and therefore it is not shown).

Figure 10: Data structure for historical parking occupation information.

The rationale for this hierarchical data structure is to always have available and up-to-date a number of historic views, which are directly built (and incrementally updated) on the basis of the incoming stream of time-stamped observations. The data is organized and stored as follows in the tree displayed in Figure 10.

- The “Key” node at the top holds the information about a single macro/micro-zone in a given year; therefore, we have a tree like this for each city micro-zone and each year.
- The “Days” part of the hierarchy organizes all the observations received for that micro-zone by day and hour. At each week day node (1 to 7) and hour node (0 to 23) there is the corresponding aggregate, that is, rolled-up information for that specific day or hour; for example, to retrieve the average parking occupation rate in a given city micro-zone on Wednesdays between 15:00 and 16:00 hours for year 2015, one has simply to navigate the “Days” sub-tree to the “4” node (for Wednesday), and then to its “15” child node.
• Attached to each of the leaf nodes (i.e. all round nodes in Figure 10) we have pointers to the all corresponding individual time-stamped observation, that is, pointers to JSON records in the format displayed in Code Listing 4.

• These records are not shown in Figure 10 for the sake of space and clarity, but they are immediately accessible for retrieval or audit.

• The “Workdays” sub-tree represents a different view on the same data in the “Days” sub-tree, specifically on the subset of observations which have come in for working days (Mondays to Fridays), which are all aggregated together.

• The “Weekend” sub-tree is analogous to the “Workdays” view, but for parking occupation rates aggregated over Saturdays and Sundays.

• The “Months” sub-tree works very much similarly to the rest of the tree, which we have just discussed above, but inserts another temporal granularity to the hierarchy, and supports by-month roll-up of the data.

This kind of hierarchical data structure incorporates and augments the “flat” data structure represented by simply the incremental log list of incoming time-stamped observations; moreover, it is quite convenient for supporting time series comparison analysis of the data recorded by those observations at different levels of granularity.

5. REAL-TIME DATA MANAGEMENT

STREETLIFE addresses the near future challenges for capturing, storing, processing and communicating increasing quantities of information, which may come from traffic management systems, connected vehicles, the surrounding infrastructure and from mobility users including floating GPS data and crowdsourcing information. In this context, our research focus is in large systems, where data management requirements are faced with the challenges of scalability and speed of access for real-time needs.

Our live environment is heterogeneous, with distributed sensory sources and varying amounts of computational and communicational capabilities. While a vehicle fleet tracking system may involve very simple data sources, which only transmit position updates for backend processing, our mobile users may possess tablets and smart phones with resources matching or even exceeding their home computers. Furthermore, our mobile users are able to participate in the transport scenario both actively and passively, producing rich data.

In STREETLIFE, our goal is to find solutions to the challenge, tackling the problems of scalability and speed. Simultaneously, we need to consider standards along with our view towards a Smart City Reference Architecture (D2.2.2 [17]). To handle these partially conflicting requirements, we separate them to internal and external parts. Internally, for the real-time data management case, we focus on the engineering challenges, algorithms and optimisations. Externally, we relax the requirements to be compatible with the external, standardised world.

In this Section, we identify and analyse our main challenges, and create efficient internal solutions.

For the architectural design of large-scale distributed interactive systems we identify two main challenges:
1. **Computational load**: Does the computational load of individual nodes or a central one increase along with the number of nodes?

2. **Network load**: Does the network load increase along with the number of connections?

If the computational load for each individual node increases as nodes are added, the maximum number of nodes in the system is then limited by the computational capabilities of the nodes or the central node. In an infinitely scalable system, computational load would be independent of the number of nodes.

Similarly, if network load increases with the number of added connections (between nodes), the maximum number of connections is limited by the network capacity. Again, if network load for any given connection is independent of the total number of connections, the networking solution could scale infinitely.

In both cases, the real world capabilities of large-scale distributed systems depend on implementations, both on hardware and software level.

In D3.2.1 [2], we discussed various early networked virtual environments (NVEs). Of these, the main challenges were first encountered and addressed in SIMNET and the follow-up NPSNET-IV military simulations. These platforms were composed of multiple different parts and were extended from computer simulations to real entities and hybrids. After several iterations of implementations, improvements and public collaborative meetings, a Distributed Interactive Simulation (DIS) environment was defined.

Distributed Interactive Simulation is a government/industry initiative to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual worlds for the simulation of highly interactive activities. This infrastructure brings together systems built for separate purposes, technologies from different eras, products from various vendors, and platforms from various services, and permits them to interoperate. DIS exercises are intended to support a mixture of virtual entities with computer-controlled behaviour (computer generated forces), virtual entities with live operators (human in-the-loop simulators), live entities (operational platforms and test and evaluation systems), and constructive entities (war-games and other automated simulations). DIS draws heavily on experience derived from the Simulator Networking (SIMNET) program developed by the Advanced Research Projects Agency (ARPA), adopting many of SIMNET’s basic concepts and heedning lessons learned.

In order for DIS to take advantage of currently installed and future simulations developed by different organizations, a means had to be found for assuring interoperability between dissimilar simulations. These means were developed in the form of industry consensus standards. The open forum (including government, industry, and academia) chosen for developing these standards was a series of semi-annual Workshops on Standards for the Interoperability of Distributed Simulations that began in 1989. The results of the workshops have been several IEEE standards. These standards provide application protocol and communication service standards to support DIS interoperability.

The interoperability components addressed by these standards and the recommended practices are:
a. Application protocols  
b. Communications  
c. Exercise management and feedback  

The standard IEEE Std 1278.1-1995 [19] defines the format and semantics of data messages, also known as Protocol Data Units (PDUs), that are exchanged between simulation applications and simulation management. The PDUs provide information concerning simulated entity states, the type of entity interactions that take place in a DIS exercise, and data for management and control of a DIS exercise. IEEE Std 1278.1-1995 also specifies the communication services to be used with each of the PDUs.

DIS is intended to support the following functional requirements:

a. Entity information/interaction  
b. Warfare  
c. Logistics  
d. Radio communications  
e. Distributed emission regeneration  
f. Simulation management  
g. Synthetic environment  
h. Entity management  
i. Minefield  
j. Live entity information/interaction  
k. Non-real time  

Of these, the entity related definitions are most relevant for STREETLIFE. Entity information exchanged between simulation applications includes the type of entity, its location, its orientation, and how the entity might appear to others. DIS requires that entities be enumerated based on their entity type, allowing a variety of different entities to be represented. Sending the location and orientation of an entity is critical for correct representation of the entity by other simulations on the network. Inclusion of the velocity and the acceleration parameters allows receiving simulations to employ higher-level, higher-accuracy extrapolation routines (dead reckoning). Multiple parameters exist to define visual appearance of an entity.

In DIS, entities can interact in multiple ways. In this context, we extend interaction to include visualization in a one-directional manner: a viewer needs entity updates in order to visualize entity states, but an entity does not need to know about a viewer. In all cases of entity interactions, the relevant state information associated with the interactions needs to be exchanged. For visualization, all parameters that affect the appearance of an entity need to be transmitted. For entity collisions, other parameters such as elasticities, masses and inertia tensors would be needed. For localized simulation, environmental conditions such as rain, snow, fog, clouds, ambient illumination etc. need to be available for each simulator. In a hybrid environment, these can be measured on site and passed to simulations.

Some of the newest additions to DIS include Live Entities. Regarding live entities, DIS considers conservation of network bandwidth to be a prime concern. Table 4 presents the improvements on
DIS network protocol to better adapt to the live situation. In addition, external, static data sets can be distributed prior to a live event.

**Table 4: Architecture modifications to DIS for supporting Live Entities**

<table>
<thead>
<tr>
<th>Modification</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce length of frequently transmitted data</td>
<td>Break Entity State PDU into multiple PDUs so that they may be transmitted at different intervals and so that slowly changing information need not be reported as frequently as quickly changing information. Reduce the size of the PDU fields by eliminating implied precision that does not match the accuracy of the data measured.</td>
</tr>
<tr>
<td>Eliminate unused or non-changing fields in PDUs</td>
<td>Some information on some platforms cannot be supplied because it is not instrumented. A field containing a set of bit flags is included in some PDUs to signal the presence or absence of specific optional fields. Use reference points for specification of LE location</td>
</tr>
</tbody>
</table>

5.1. **A lightweight networking implementation**

In STREETLIFE, our real-time networking protocol bears similarities to DIS. DIS has been developed from pragmatic requirements, and does not support XML or other bloated specifications; instead, it is a compact binary representation, with special optimizations for live data. Similarly, our protocol is biased toward efficiency. It is defined with XML, but actual Protocol Data Units (PDUs) are transformed from that into binary, tokenized XML expressions. The networking implementation is targeted at the mixed reality App, which provides a direct visualisation of the data with very low latency requirements.

Code Listing 5 presents the current entity PDU used by the mixed reality Apps.(D5.2.2 [5]) This message is used to update any entity state in the STREETLIFE system. Conformant with DIS’s Live Entity modifications to the main DIS protocol, only fields id and flags are mandatory, and the rest are optional and their presence in the PDU defined by flags. Also `road_position` represents a local coordinate system within a current street segment in our World Graph, and can replace the global `position`. We use `gp_string` as a general purpose string to extend our protocol with new, temporary fields.

Each field can have its own extrapolator, or *dead reckoning* method, defined in the `reckonings container` holding a list of necessary parameters and the related algorithm. The current parameters are presented in Code Listing 6. In a typical case, they would be transmitted only once, or pre-defined. We currently support three simple extrapolation methods: 1st degree, 2nd degree, and *constant trigger*. The use of extrapolation is similar to NPSNET IV’s *Ghost/Player* paradigm: as long as a measured value is within allowed distance from an extrapolated one, it does not need to be updated. Use of the constant trigger is contextual: for example, it could represent the current street segment. When a new segment is reached, an update
would be necessary. Or, it could represent a count of segments, allowing an entity to move $N$
segments before an update is needed.

```xml
<struct name="entity_message">
  <header>
    <field name="id" type="uint32_t"/>
    <field name="flags" type="uint32_t"/>
    <field name="type" type="uint32_t"/>
    <field name="privacy" type="privacy_type"/>
    <field name="name" type="string"/>
    <field name="owner" type="string"/>
    <field name="target" type="string"/>
    <field name="frame" type="uint32_t"/>
    <field name="time" type="time_t"/>
    <field name="position" type="coordinates"/>
    <field name="road_position" type="road_coordinates"/>
    <field name="course" type="coordinates"/>
    <field name="visual_tile" type="visual_tile"/>
    <field name="visual_mesh" type="uint16_t"/>
    <field name="photo" type="uint32_t"/>
    <container name="reckonings" type="dead_reckoning"/>
    <container name="links" type="bullhead_message"/>
    <container name="notifications" type="notification"/>
    <field name="occluded_mesh" type="uint16_t"/>
    <field name="gp_string" type="string"/>
    <field name="heart_rate" type="uint8_t"/>
    <field name="noise_level" type="uint16_t"/>
    <field name="light_level" type="uint16_t"/>
  </header>
</struct>
```

**Code Listing 5: A lightweight entity update PDU definition**

```xml
<enum name="dead_reckoning_type">
  <option name="first_degree"/>
  <option name="second_degree"/>
  <option name="trigger_constant"/>
</enum>

<struct name="dead_reckoning">
  <field name="type" type="uint8_t"/>
  <field name="velocity" type="coordinates"/>
  <field name="acceleration" type="coordinates"/>
  <field name="tick" type="time_t"/>
  <field name="threshold" type="float"/>
  <field name="trigger_id" type="uint32_t"/>
  <field name="field_id" type="uint32_t"/>
</struct>
```

**Code Listing 6: Entity state dead reckoning parameter**
Bindings for the network protocol are automatically generated for Java, C and C++. The Activity Recognition App uses only Java, while the Mixed Reality app uses C++. At network level, the PDUs are the same. While the PDU definition does not address the carrier, we currently use TCP/IP. Given TCP’s high overhead due to large headers, in future we may add support for UDP/IP.

5.2. Interest Management

In both, the Mixed Reality and Activity Recognition App, entity data is published and transmitted to a server when dead reckoned states exceed given error margins. From the server, it is propagated to any interested clients. We support implicit and explicit interest expressions. For example, a client can request a data feed from an individual entity (such as a friend), or it can assume an interest management (IM) scheme, and publish its own information, relying on the server to determine if updates from any entities are needed.

Here, interest management acts as a means to minimize network transmissions between entities (and clients, which in our case are the same) to only those that are needed. We foresee cases such as:

1. Traffic state visualisation on a rectangular area such as a map interface.
2. Travel assistance with interest in events that can affect the route.
3. 3D Live environments with a free viewpoint in 3D space (Mixed Reality).
4. Sensor data propagation for online spatial analysis (can include very short-term events like car honking).

In the following, we provide examples of the first two cases.

Traffic state visualisation on 2D interface: Assume near future large-scale anonymous traffic tracking, where vehicles and pedestrians allow them to be tracked, and send their state updates to a tracking server. For real-time visualisation of the full traffic state on a map, an operator defines a rectangular area either explicitly or by simply zooming closer (Figure 11). Each street segment that lies fully or partially within the area is now activated; the operator has subscribed for updates. This selection is communicated to the server either as a list of street segments, or by corner coordinates, in which case the server expands it into a list of segments. Now any update that takes place in a given street segment is immediately passed to the operator, and visualised. Operations at the tracking server involve only in-memory street segment table updates and subscription list management.
Figure 11: Defining a two-dimensional interest expression. All street segments within a rectangular viewing area are subscribed for updates.

Travel assistance with real time event propagation: Assume a traveller who has just made a route request from A to B (Figure 12). The route is defined as a set of street segments. This forms the primary interest expression. To avoid any surprises, our traveller wants to receive all events that may have an effect on her journey; traffic jams, accidents and other anomalies. As the street network is connected, any event near the route may be important. We now implicitly define a 1-hop neighbourhood of street segments from each primary route segment. This can be formed either at client side or server side, with hop count as a parameter. These segments are now active and events on them are passed to the traveller. In Figure 12, a traffic jam is causing a bottleneck on the route, and this is passed to the traveller. Her router is then suggesting an alternate route.
Figure 12: Topological interest expression by defining a route with a 1-hop neighborhood

Similar segment subscription mechanisms are used for defining a visibility based interest expression for 3D live maps, using a pre-defined visibility look-up list for each view point in a 3-dimensional grid. For sensor data, such data is linked to the underlying street segments, and triggers can be set for exceeded thresholds.

All these schemes rely on look-up lists of subscriptions, resident in memory. As no complex online calculations are performed, computational load on the server does not increase significantly for each additional connection.

We have currently implemented two implicit interest management schemes: one for a given rectangular area and one for potentially visible entities in a 3D scene. The visibility based IM is currently limited, and not yet based on large-scale map data. Please refer to D5.2.2 [5], Section 5 for details.

The increase in networking load depends on the frequency of entity updates, the size of used entity update PDUs, and the spatial density. In addition, for the visibility based IM, there are cases where the look-up table results in almost all-to-all transmissions. For example, if all viewing clients of the mixed reality App were high in the air, they would see all vehicles and users moving on a road network. Currently, we do not support visualization of viewpoints themselves, so we don’t provide such entity updates. This limits transmissions for this worst-case situation from entities to viewers.
6. **MODE DETECTION**

6.1. **Introduction**

Mode detection is required in STREETLIFE to identify, which transport modes are used and what are thus transport-related emissions. Only when the utilised modes are known it is possible to attempt a modal shift towards more sustainable transport modes through incentivization and gamification. As already mentioned in the context of user tracking, mode detection is also required in STREETLIFE on a user level, to provide real-time support. Crowdsourcing and Route deviation recognition are two components that can directly benefit from the mode detection functionality.

The idea of mode detection is to recognise automatically the transport mode a STREETLIFE user has selected instead of deciding the mode by user input. Transport mode detection is an important field of transport research belonging to a more general activity detection research field that has become more important with the higher penetration of smartphones and embedded sensors. Algorithmic mode detection from sensor data is not a trivial problem although the mode is easily recognisable by a human being.

Smartphones can have multiple sensors presented in Table 5. Some of them are not useful in mode detection.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Relevance to mode detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>Location information can be used to calculate speed and filter out certain modes. Location information can also be used to map against cycle paths, tram lines, railways etc.</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>Most of mode detection algorithms are based on acceleration data.</td>
</tr>
<tr>
<td>Gyroscope</td>
<td>Gyroscope data can be used for detection of modes, where the rotation of the mobile phone varies.</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>Magnetometer data may give distracted output in magnetic fields. Rail modes may affect magnetometer and thus be used for rail mode detection.</td>
</tr>
<tr>
<td>Heart rate monitor</td>
<td>Typically heart rate is different between motorised and non-motorised transport modes and can be used to distinguish between them.</td>
</tr>
<tr>
<td>Thermometer</td>
<td>Inside and outside temperatures vary, but inside temperature can be kept constant. If the temperature is different in different motorised vehicles it can be used for distinguishing modes from each other</td>
</tr>
<tr>
<td>Barometer</td>
<td>The atmospheric pressure varies according to the altitude. This has of little use in ground transport mode detection, but can be used to distinguish air traffic transport mode from the others.</td>
</tr>
</tbody>
</table>
The end result of a mode detection algorithm is information on when and where the user (mobile device owner / holder) has used different transport modes. The information can be stored in tabular format. It can have regular time intervals with the transport modes or the algorithm can give only the time (and location) when the mode changes.

6.2. Mode detection approaches

There are two important decision criteria in choosing the mode detection approach. First, the actual algorithm can run either on the Client or Server side. Secondly, the latency between the actual activity and the computed result make mode detection either real-time or post-journey computation. The decision space is shown in Figure 13.

Since mode detection algorithms can resort to complex (and slow) machine learning methods computational power may be needed and therefore a server side solution is advantageous if high accuracy is aimed for. The disadvantage is increased data transfer of the sensor data from the Client to the Server. As a rule-of-a-thumb higher the accuracy required - more sensor data needs to be transferred. In European context and urban areas this is not a major problem anymore, but in rural areas the network coverage may be poor, the bandwidth low and especially in developing countries the data transfer price is still high. In this case the sensor data needs to be stored in the memory of the mobile device and data would be transferred once feasible. Battery consumption is another criterion to be taken into consideration since it is sensor-dependent and data transfer consumes battery, too.

![Figure 13: Decision space for mode detection](image)

In terms of latency it is possible to implement a near real-time mode detection if the computation is done either on the Client side or if there is constant and fast data transfer from the Client to the Server. However, there is a general trade-off between the latency and accuracy since
accurate algorithm generally relies on more complex algorithms and / or more data, which requires more computational power and the shift of the computation is towards the server side.

Another important factor in mode detection implementation is possible data fusion with external data sources besides the actual Client device. Geographic Information on public transport routes and transport infrastructure is the most imminent data source available on OpenStreetMap, for example [20]. Other external data sources could include number plate recognition of vehicles at traffic count points or parking places or user-specific Social Media input.

6.3. Initial requirements

Incentivation for greener transport implemented by Gamification requires information of the transport modes utilised by the users. Therefore, a decent accuracy of the percentage of each transport mode is required. Altogether, there are six modes that need to be recognised: by foot, passenger car, bus, tram, bicycle and train.

Decent accuracy is debatable. Minimisation of random error instead of systematic error is more important in terms of incentivisation. Also, total emissions by STREETLIFE users calculated in the emission control panel can be rectified if systematic error is known. The minimum accuracy for each transport mode should be 90%.

Incentivation and emission calculations can be performed after the trip and therefore the calculations can be done on the server side and not real-time. However, Client side real-time mode detection would be beneficial for detection of real-time traffic situation that could be visualised by 3D models. Another major motivation for mode detection at the Client side is scalability in the large scale. As the current mobile devices have major computational power parallel computation for mode detection is more feasible to distribute the computations at least partly to the mobile devices in case there are tens of thousands of users.

6.4. Selected strategy for mode detection

Mode detection approach in STREETLIFE is based on incremental increase in accuracy and functionalities. The approach is also hybrid in the sense that not all computations are one either at the Server or at the Client side, but in both. The initial mode detection algorithm is based on the existing ActivityRecognition API [21] of Google Play services that uses sensor data of Android devices and produces probabilities for the current transport mode. The data logger produces the following information:

- Timestamp
- Actual Activity
- Latitude
- Longitude
- Probability for the user remaining STILL
- Probability for the user being ON FOOT
• Probability for the user being WALKING
• Probability for the user being RUNNING
• Probability for the user being ON BICYCLE
• Probability for the user being IN VEHICLE
• TILTING

Since Google mode detection component fails to distinguish between buses and passenger cars as well as trains and trams from other transport modes, the enhanced version is implemented by post-processing the collected data with the help of external data sets and tracking data. The schematic outline of the process is shown in Figure 14.

- Figure 14: Mode detection with Google Play services and post-processing

6.5. Initial results

Google Play services were tested in Espoo, Finland in a route that included walking, running, passenger car driving and cycling. The actual route is shown in Figure 15.
The total duration for the route was around 50 minutes. Table 6 shows the number of observations per activity by the detected mode and the actual mode.

### Table 6: Detected and actual modes

<table>
<thead>
<tr>
<th>Activity</th>
<th>Number of detected observations</th>
<th>Number of actual transport</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>154</td>
<td>149</td>
<td>93%</td>
</tr>
<tr>
<td>Running</td>
<td>41</td>
<td>84</td>
<td>48%</td>
</tr>
<tr>
<td>Cycling</td>
<td>38</td>
<td>38</td>
<td>89%</td>
</tr>
<tr>
<td>In a vehicle</td>
<td>214</td>
<td>230</td>
<td>90%</td>
</tr>
</tbody>
</table>

### 6.6. Future improvements

The most important improvement for mode detection by post-processing is detection of bus, train and tram travelling. These can be detected by using external data source for tram and train track as well as bus stop locations. First, labelled multi-modal route data has to be collected and identified which modes are confused with the rail transport. If the confusion is not systematic the
tracking data (GPS coordinates of the user’s location) are matched with the rail line locations in the following way:

1. Loop for the whole track data
2. Calculate the minimum distance to any train station
3. Calculate the minimum distance to any tram station
4. Calculate the minimum distance to any train track
5. Calculate the minimum distance to any tram track
6. If the minimum distance to rail track < threshold between points A and B AND the minimum distance from the rail stations to the points A and B < threshold AND track and station = same mode (train or tram) the leg = rail mode

In case of bus transport similar matching is conducted, but using only the data that has been classified as vehicle transport.

In terms of increasing accuracy of detection of all the transport modes the following methods will be tested:

- k-NN (Nearest Neighbour)
- Support Vector Machines (SVM)

The first method is fast and simple and could be used to rectify some erroneous classifications of the Google Play services. If k-NN does not tackle all the cases, SVM could handle a more complex classification problem [22]. In this case the data would be divided into training and validation sets and cross-validation will be used to train with randomly selected training data. The selected features for the hyper plane are the probabilities for each activity estimated by Google Play services as well as historical values for \( n \) time steps. In this way, with labelled data it is possible to get rid of single points erroneous classification.

According to the findings so far, the Google Play services produce occasional error points that could be rectified using the afore-mentioned methods and sensor data for training.

7. **BICYCLE SAFE ROUTES**

The development of safe bicycle routing in the Berlin pilot is realized through the process of three different steps. During the first phase the ground grid was chosen and needed to be extended by several new attributes, playing a role for cycling safety. In a second step an accident analysis was carried out to prepare the subsequent third step: developing the new cycling safety bike router.

The VMZ Berlin maintains and updates in regular intervals the official street network for the Senate Department of Urban Development and Environmental Protection of Berlin – the so called “Berlin Detail Network”. In the above mentioned first step, this network was extended by cycling relevant information and attributes. Especially details about surfaces of the underground and the quality of the surfaces have been added as new attributes in the tool MapInfo, being a
Geographic Information System (GIS). The following list shows all main attributes, which were added to the network.

- Street classification according to StEP
- Street classification according to RAS06
- Predominant use of the surrounded buildings
- Centre strip
- Type of centre strip
- Cycling guidance
- Roadside parking traffic
- Parking management
- Bus
- Separate bus lanes
- Tram
- Position of tram
- Type of intersection
- Pedestrian crossing point
- Crosswalk
- Street name
- Length of section
- Number of lanes
- Traffic control
- Bicycle Paths

At the end of phase one a quality check secures all content and information.

Thereafter, the second step could start: The accident analysis was conducted through a VMZ own tool called ProVista. As a result of a national research project, the ProVista client allows to execute extensive evaluations of accidents. To do so the official accidents stated by the Berlin Police Department for the period of January 2010 and August 2014 were checked and evaluated. The accidents were referenced to the Berlin Detail network. After that, the accident data were evaluated according to official procedures and methods: Specifically, the following guidelines have been considered and included for analysis of accidents: To determine accident accumulations the FGSV\textsuperscript{1} standards and the UDV-Limits\textsuperscript{2} for personal injury as well as heavy personal injury including cycling modes were used. The results of this investigation are accident black spots and accident rates.

All official accident spots which include personal injury (light and heavy) on road sections and intersection were identified in the above mentioned period (January 2010 and August 2014). In addition all accidents with cycling modes which include personal injury (light and heavy) and property damage were identified as well. Furthermore, all accidents involving cyclists have been identified. These accident spots and these accident black spots have been superimposed to results of the Berlin Senate survey. In this survey the public was asked to state their personal unsecure hotspots.

---

\textsuperscript{1} The Research Society for Roads and Transportation (FGSV) is a non-profit technical and scientific association.

\textsuperscript{2} The UDV (German Insurers Accident Research) is part of the German Insurance Association (GDV)
In addition to the accident spots all accidents including cyclists were analyzed regarding to weather (rain, ice, snow and blending sun) and infrastructure (parking traffic, stops and stations and cycling guidance) as the cause of the accident.

As one result the following two figures (Figure 16 and Figure 17) show exemplary accidents at links and intersections involving the cyclists for the year 2013 in Berlin.

Figure 16: Accidents with cyclist at links for the year 2013
Figure 17: Accidents with cyclist at intersection for the year 2013

In addition Table 7 sums up all accidents and the distinction between the damage for 2013.

Table 7: Accidents with cyclist at intersection for the year 2013

<table>
<thead>
<tr>
<th>Category</th>
<th>Amount</th>
<th>Personal injury and property Damage</th>
<th>Property Damage</th>
<th>Personal Injury</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclists Participation</td>
<td>3.439</td>
<td>2.420</td>
<td>1.468</td>
<td></td>
</tr>
<tr>
<td>All Accidents</td>
<td>63.114</td>
<td>6.976</td>
<td>906</td>
<td></td>
</tr>
</tbody>
</table>

The data from the second step will be integrated into the specialised bicycle router as a part of the Intermodal Router at the Berlin pilot site. The based network, which the router uses for its calculations, was extended by the accident black spots at nodes and links. The algorithm is able to take into account them when calculating a route and avoid them. For each routing request the avoidance can be explicitly turned on and off. But the functionality takes more than only the hot spots into account.
Table 8 lists three cycling preferences, which will be integrated in the App frontend that the user can easily switch and change between all options.

**Table 8: Cycling preferences**

<table>
<thead>
<tr>
<th>Fast Route</th>
<th>Safe Route</th>
<th>Comfort Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>The route is suitable for racing bicycles and uses esp. very good road infrastructure</td>
<td>The route avoids accident black spots and associated infrastructure elements</td>
<td>Route avoids main roads and cobblestones</td>
</tr>
</tbody>
</table>

**It will take into account the following criteria:**

<table>
<thead>
<tr>
<th>Avoid bad surfaces</th>
<th>Analyzed avoid black spots</th>
<th>Avoiding main roads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid traffic lights</td>
<td>Avoid subjective problem priorities</td>
<td>Avoidance of cobblestones</td>
</tr>
<tr>
<td>Avoid intersections</td>
<td>Cycling guidance (bicycle road, bike path stripes, etc.)</td>
<td>Avoiding bad state of the cycle track surfaces</td>
</tr>
</tbody>
</table>

| Cycling guidance (bicycle road, bike path / strips, etc.) | Cycling guidance (bicycle road, bike path / strips, etc.) | Cycling guidance (bicycle road, bike path / strips, etc.) |
8. Conclusion and Future Work

This deliverable shows the progress we have made in advancing our models and techniques for mobility data integration and management. We have elaborated on our work through the main key points that defined the work of WP3 in Y2.

We have elaborated on the new STREETLIFE crowdsensing and crowdsourcing initiatives. In this deliverable we have shown the conceptual and implementation work we have done on these two concepts. Even though the concepts are mode independent, we have concentrated on bicycles, as we see the research and development for this transportation mode brings a high added value to the STREETLIFE objectives.

These two initiatives are going to be part of the second phase of the pilot tests and the data collected at the pilots is going to be basis for our Y3 development. We are going to evaluate the crowdsourcing approach and identify other scenarios and applications, for expanded usage of this proactive way of collecting user-generated information. We plan to also integrate crowdsourcing with Gamification to further encourage engagement of citizens towards actively producing user-generated information that is valuable for the smart mobility system.

Bicycle crowdsensing data will be collected for bicycle route generation and evaluation. Techniques for enriching cycling routes with crowd-generated content will be also evaluated and further developed.

The Real time data management has remained one or our main focuses in Y2 as well. We have shown how we advanced the algorithms for real-time data and interest management. In Y3 we will concentrate on measurement of performance and scalability of the algorithms. This will be done by simulating with 10 000 pedestrians who receive information from the interest management engine.

World graph will be further developed to include more data sets and the feasibility of World Graph with all the pilots will be assessed.

In this deliverable we have also set the basis for mode detection. We have developed simple mode detection functionality and integrated it in the existing apps. In Y3 mode detection algorithm will be further developed to include post-processing with machine learning algorithms and external GIS data. The accuracies will be tested with the user data collected from the city pilots. For GIS data utilisation data sources will be identified and data extraction algorithms developed to retrieve tram and train tracks as well as bus lines and bus stop data.

The most important task for WP3 in Y3 will be learning from the evaluation and using the lessons learned to improve and solidify our innovations into exploitable technologies, in view of the project Milestone MS6 at the end of project.
APPENDIX A: LITERATURE


