Ref. Ares(2014)3660163 - 04/11/2014

FP7-SMARTCITIES-2013

STREETLIFE

Steering towards Green and Perceptive Mobility of the Future

WP3 - Mobility Data Integration

D3.2.1 – Mobility Data Integration Models and Techniques (initial)

Due date: 30.09.2014 **Delivery Date**: 09.10.2014

Author(s): Monika Mitrevska (DFKI), Giuseppe Valetto (FBK), Antti Nurminen (AALTO), Mika Vuorio (CGI),Daniel Schal(Siemens)

Partner(s): DFKI, FBK, AALTO, CGI, Siemens

Editor: Monika Mitrevska (DFKI)

Lead Beneficiary of Deliverable: DFKI

Dissemination level: Public **Nature of the Deliverable:** Report

Internal Reviewers: Birgit Obst (Siemens), Norbert Reithinger (DFKI), Jens Kohlmorgen (Fraunhofer)

EXECUTIVE SUMMARY

The objective of Deliverable 3.2.1 “Mobility Data Integration Models and Techniques” is to present the progress and the future work planned for the next phases of the project in the five most relevant topics for Mobility Data Integration (WP3): data definition, knowledge representation, data correlation, real-time data management and crowdsourcing initiatives.

As part of the data definition task, the deliverable discusses the second iteration of the STREETLIFE data model definition. In this iteration we improve the initial conceptual model and we categorize data into real-time, dynamic and static depending on the update interval and required latency. We furthermore distinguish STREETLIFE data and External data depending on the data source.

We define general architecture for exchange, handling and storing of data belonging to the STREETLIFE data model. We further elaborate on the design of the data management subsystem for the three pilot sites: Berlin, Rovereto and Tampere.

Using the orthogonal dimensions Static vs. Dynamic and Active vs. Passive we define three types of crowdsourcing that are going to be part of the STREETLIFE crowdsourcing initiative: Active and Static, Active and Dynamic and Passive and Dynamic crowdsourcing. We report in further detail the usage of these types of crowdsourcing, focusing on what data they generate and how that data is used within the STREETLIFE mobility information system.

This deliverable gives an overview on the different aspects of real-time data management. We define the real-time data source and put an accent on real-time tracking and real-time event handling. We give and overview of the related work and describe the on-going research on real-time data management in STREETLIFE.

As part of the data correlation and analysis task, we present the initial, planning phase of personalisation in STREETLIFE based on static and real-time data. We present the main correlation possibilities and a platform on which we plan to develop the data correlation and analysis component.

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.

Mobility Data Integration Models and Techniques (initial)

Table of Contents

ABBREVIATIONS

CO	Confidential, only for members of the Consortium (including the Commission Services)
BER	STREETLIFE Berlin-Pilot
CIP	City Intelligence Platform
D	Deliverable
DATEX II	Road Traffic Data (CEN/TC 278/WG8)
DoW	Description of Work
DS	Data Source
FP7	Seventh Framework Programme
FLOSS	Free/Libre Open Source Software
GTFS	General Transit Feed Specification
GUI	Graphical User Interface
IMP	Siemens Integrated Mobility Platform
IMR	Intermodal Route
IPR	Intellectual Property Rights
KPI	Key Performance Indicator
MGT	Management
MS	Milestone
OS	Open Source
OSM	Open Street map
OSS	Open Source Software
O	Other
P	Prototype
PBF	Protocolbuffer Binary Format
PI	Performance Indicator
PU	Public
PM	Person Month

R	Report
ROV	STREETLIFE Rovereto-Pilot
RTD	Research and Development
RTPIS	Real time passenger information system
SDM	STREETLIFE Data Model
SIRI	Service Interface for Real Time Information (CEN/TC 278/WG3)
TRE	STREETLIFE Tampere-Pilot
WP	Work Package
Y1	Year 1

PARTNER

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

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

WP3 as a main data integration package in STREETLIFE works on different aspects of data management in the multimodal transportation domain, using and building on existing technologies like Future Internet and Cloud Computing. The work in the package is organized in five tasks: Data definition, Real-time data, Crowdsourcing, Intermodal mobility data representation and Data correlation and analysis. This deliverable gives an overview on the progress of the tasks within the first year of the project.

We start the deliverable by presenting the STREETLIFE data model as continuation of the work done in the Deliverable D3.1[1]. Chapter 2 describes the revision process of the model and the changes that were done as a result of this revision. In STREETLIFE we employ a complex combination of different data types that require different management techniques. To set the basis for the data management we categorize the data by several criteria. The categories and their definitions are also presented in this chapter.

The STREETLIFE knowledge representation is discussed in Chapter 3. We first give a general overview and describe the architectural placement of the data management subsystem. Different subsections describe the knowledge representation and integration of STREETLIFE data and data coming from other domains in the three pilot sites.

Crowdsourced data will be researched as potential data source for a multimodal route planner and assistant system. We will explore the possibilities of different crowdsourcing techniques, aiming to define a reliable service that can contribute to the multimodal transportation system. Chapter 4 gives an overview of the crowdsourcing initiative within STREETLIFE. Here we define three categories of crowdsourced data and we present the STREETLIFE engagement plan with each of them.

One of the main objectives of WP3 is the development of techniques for fast and scalable management of real-time data. Definition and elaboration on different aspects of real-time data in STREETLIFE is given in Chapter 5. We identify the requirements by each of the aspects and provide one reference implementation of the techniques developed within the project.

By utilizing Open Data, STREETLIFE demonstrates one of the benefits of having the effort of creating such data sources. Municipalities are able to see how STREETLIFE Use Cases bring the data to life and encourage the public for greener means of travel, and witness how the STREETLIFE Pilots bring out new services for local travellers. Open data and the formats available for open data are discussed in Chapter 6.

Data correlation techniques and algorithms will be also researched to improve the context awareness of the system and to provide more relevant input for personalization and user support. In Chapter 7 we discuss the initial investigation results, suggesting a platform on which solutions will be developed, and a plan for its realization.

2. DATA DEFINITION - STREETLIFE DATA MODEL (SDM)

This chapter describes the data definition process in STREETLIFE.

Deliverable 3.1[1] elaborated on the process of data collection and data model definition. The data available at the pilot sites was collected, analysed against the requirements of the pilot scenarios and captured into the conceptual STREETLIFE data model (SDM). During the first year of the project the activities in the rest of the work packages have influenced the data requirements as well. Section 2.1 describes the first revision and changes done on the SDM.

Bringing the model closer to realization, Section 2.2 views and categorizes the data from two different perspectives. The categories are defined in respect to the update interval, required latency, and the origin of the data.

2.1. Phase II: SDM Revision

In the first definition phase of the SDM the accent was put on the external data that is already available and used at the pilot sites. The changes from the first revision concern the data produced by STREETLIFE services. Partially this data was included in the initial model, but as the work in the other packages was progressing, the model needed to be refined.

The requirements for the first revision of the data model have come from WP4, WP5 and WP8.

WP4 defined the functionality of the STREETLIFE Control Panel. Main concepts used in the work of WP4 are *Context* and *Transportation* (Table 2). After analysis of the available data (Deliverable D4.2.1[2], Data model), it was realized that the current version of SDM does not capture the modal split. In its first version, SDM defines current transportation mode of the user as motion type (as part of the *TravelContext*). Modal split is an aggregated STREETLIFE data that captures the usage of different travel modalities (e.g., car, bicycle, public car, shared car). This value can be defined in the context of a *User*, an *Event (cityEvent)*, a *Trip* or a *Service*. Modal split was added to the SDM and linked with the relevant concepts.

WP5 collaborated with WP3 in defining the requirements for the STREETLIFE end-user application. The data model was considered mainly from the end-user perspective, making sure that SDM captures all the user data required in STREETLIFE. Refinement of the model has been done in two concepts: the *User* and *Services*, in particular the route planning service.

STREETLIFE defines different user types based on the anonymity level which were also added to the model.

After collecting and analysing the key performance indicators (KPI) relevant for the pilot sites (D8.1.1 [3]) WP3 identified the data that was not included in the data model and that is relevant for KPI calculation.

The PIs that required an update of the SDM are shown in the Table 1. The complete table of PIs is presented in D8.1.1[3].

Table 1: Performance indicators considered for SDM update

PI	Indicator	Unit	Required measures	SDM reflection
PI-102	Average modal split – trip purpose	Percent	a) trip purpose b) mode of transport	SDM: *Trip* Trip purpose
PI-103	Number of Stops	Integer		SDM:*Tip*: Number of stops
PI-104	Sharing bikes placed near an event	Integer	a) # of sharing bikes b) location of sharing bikes	SDM: *BikeSharing* log for shared bikes
P-108	Average speed per trip	km/h	a) start time b) end time c) distance	SDM:*Trip*: Distance
PI-314	Number of shared STREETLIFE trips			"Shared STREETLIFE trips" needs to be defined
PI-401	Number of bike accidents			The availability of this data needs to be confirmed. At the moment added as an event that can be reported by users.
PI-403	Number of selected safe cycle routes	Integer		"Safe cycle route" needs to be defined within STREETLIFE.

Table 2 provides an overview of the main concepts of the STREETLIFE data model. This model should be seen as a complete working version but not as the final version of the model. Further development of the project and implementations might define changes to the proposed data which will trigger refinement and detailing of the data model.

Table 2: STREETLIFE Data Model - Main concepts

Main concepts		Classes
User	Authentication	LoginData, UserTypes
	Basic Information	Abilities, ContactInformation, Demographics
	Devices	UserDevice, SystemPreferences, UserVehicles

	Relationships	UserRelationship, RelationshipTypes
	STREETLIFEProfile	UserServices, MobiloityPreferences, RoutingPreferences, TripHistory, RouteChangeTrack, CommonTrips, FavoriteTrips, UserGreenProfile, UserCurrentState, UserStreetlifeProfile
Context	EnvironmentalData	Weather, CO2Emission, PollutionState
	Events	Event, EventType, SourceType
	PointOfInterest	ParkingFacility, BikeParking, VehicleParking, ParkingMeter, SharingFacilities, TrafficLights, WiFiHotSpot
	Infrastructure	Amenity, Places, Buildings, Highways
Transportation	TimeTable	
	TransportationModality	Vehicle, PublicTransport
	Network	Trips, Routes, Stops
Services	TripPlanner	RouteRequest, Trip, TripSegment, ReroutingLog
	BikeSharing	Bike, Availability, Reservations, Stations
	ParkAndRide	TBD
	ExternalServices	TBD
Communities	CrowdSourcing	TBD

Detail overview of three main concepts of the SDM at this stage is given in Figure 1. Here we present the *User*, *Events* and *TripPlanner* and the relationships between them. Further development of the model will be done for the STREETLIFE data (definition provided in 2.2.2). The part of the model that is classified as external data will not be elaborated in detail. The integration of this data is done by each of the STREETLIFE components and can be different for each of the pilot sites. The rest of the *Services* and the *Communities* model will be developed in details in the next phase of the project.

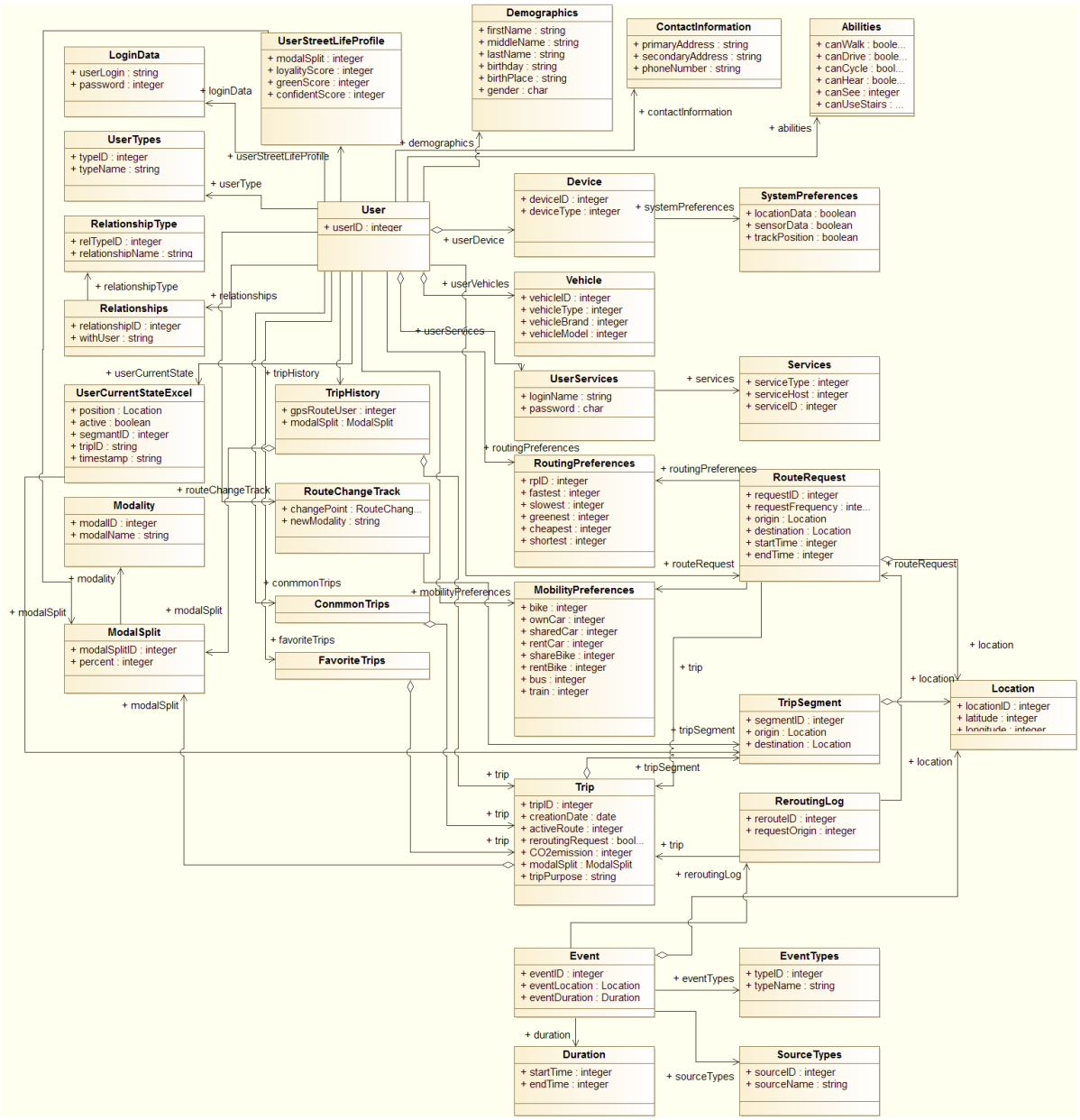

Figure 1: User model

2.2. Classification of Mobility Data in STREETLIFE

In this section we categorize the SDM data based on different criteria. In the first subsection we define four categories based on the update interval of the data and the required latency. In the second subsection we define two categories for SDM data based on the origin of the data. For classification purposes, we conceptually describe the roles of different components that access the data; we do not define the information flow between components. The information flow is described as part of the blue print architecture in D2.2.1 [4].

2.2.1. Update interval and update relevance

In STREETLIFE, we have previously categorized our data to three classes (D3.1/2.2 [1]). We considered data that has less than one second update intervals to be real-time, those between one second and one day to be dynamic and slower to be static. However, even within STREETLIFE, tasks and applications vary, reflecting variations in their real-time performance requirements. Due to application differences and requirements, this categorization needs to be considered initial and *hard*, suited for the most demanding cases, where also the process and transmission delays are expected to be within this time frame.

We will generally consider real-time data to be any *temporal content update that takes place during execution of any STREETLIFE service, and that has an immediate effect on the output or behaviour of that service.* The emphasis here is in the online data management process and effect on output. Consequently, integration of real-time data involves always on-line processing via network interfaces.

Putting the emphasis on the online management process adds the latency of the data as a second dimension in the data categorization. While real-time data is defined as data that requires low latency, the STREETLIFE system accepts a relaxed latency for the non-real-time data.

If we define now three levels of update interval frequency – high, medium and low frequency and two levels of latency – low latency and relaxed latency, Table 3 shows the matrix of data categorization into real-time, dynamic and static.

Table 3: SDM Categories - real-time, dynamic and static

<table>
<tr><th></th><th>Low latency</th><th>Relaxed latency</th></tr>
<tr><td>High frequency</td><td rowspan="3">Real-time data</td><td rowspan="2">Dynamic Data</td></tr>
<tr><td>Medium frequency</td></tr>
<tr><td>Low frequency</td><td>Static Data</td></tr>
</table>

Static data is defined as data that has low frequency updates and can have a relaxed latency within the STREETLIFE system. The content of this data is not updated, or the updates do not influence the performance of the system and does not need to be tracked. Examples of static data are: user authentication, user basic information, devices, infrastructure, basic information about services and vehicle etc.

Dynamic data can have either high or medium update frequency but the effect of the updates is not critical for the running services, therefore it can have a relaxed latency. One example of medium-frequency dynamic data is the number of available parking places. The value of this data can change over time but it is not have an immediate effect on the running services at present time.

Static and dynamic data are both part of a more general category – non-real-time data. As defined in D2.2.1: *Non-real-time data is either static data or data whose content update does not have an immediate, online effect on the output of the services*. This data can be requested offline or during service execution.

Both, real-time and non-real-time categories refer to present data. To make dynamic data available for further analysis and KPI calculation we add another category to the data model – the *historical data*. Historical data keeps track of the past content updates of real-time and non-real-time data.

2.2.2. Data origin

In the STREETLIFE system we identify two levels for data access:

Backend: data accessed by the STREETLIFE backend services

Frontend: data accessed by the STREETLIFE End-user Application and the STREETLIFE Control Panel.

To better define the data that will be accessible for each of the groups, we introduce another data categorization: by origin.

As already discussed in Deliverable D3.1[1] and D2.1[5], STREETLIFE services bring together heterogeneous mobility-related data sources. The standards and the available formats for the external data sources that are going to be used in STREETLIFE are given in D3.1/Appendix A.

In STREETLIFE we can divide the data sources into two main groups: *STREETLIFE data sources* and *External data sources*.

As *STREETLIFE data* we define the data produced and managed by the STREETLIFE services and applications. The STREETLIFE data management system is responsible for providing access to this data to the backend services and frontend applications.

The data that is produced outside the STREETLIFE system we define as external STREETLIFE data (or external data). The sources for this data might be different for different pilot sites but they usually provide the data in a standardized format with well-defined APIs to access it. Open data, as major part of the external STREETLIFE data, plays a very important role for STREETLIFE. Chapter 6 gives an example of the standard formats for Open Data. The SDM captures this data, but does not require from the data management system to provide centralized access to it. The integration of this data will be done individually by the services or frontend components that need to use it.

Figure 2 illustrates the external data sources (including the legacy data sources) as data providers and the two main groups of components: backend and frontend services as STREETLIFE data providers and STREETLIFE data and external data consumers.

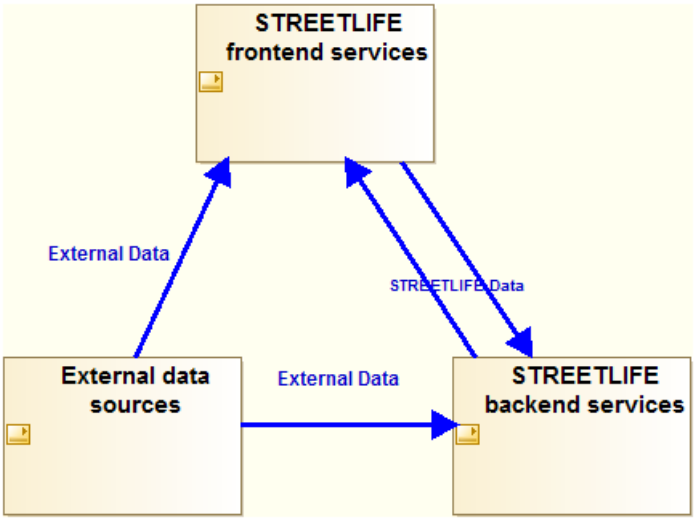

Figure 2: Data flow between frontend and backend STREETLIFE services and external data sources

Please note that some of the data might have its external and internal version. One example is available parking spaces. The park and ride service receives *external* information about available parking spaces in the city. After processing this data according to the current context, the service provides available parking places, now as STREETLIFE data.

Table 4 gives an overview of the data groups that classify into the two categories.

Table 4: SDM categories: STREETLIFE data and External data

STREETLIFE Data	External Data
User data	Maps and cartography
Service data	Points of interest
Road Events	City Events
	Public transport information
	Environmental information

At this point in the project we recognize three main STREETLIFE data groups:
User data – The user data is completely produced by the STREETLIFE services. STREETLIFE components that work with this data should provide interfaces for requesting and providing this data.
Service data – All STREETLIFE services will provide the data to the rest of the STREETLIFE system.
Events – There are two different event types in STREETLIFE: road events and city events. City events include cultural events, sport events, concerts and all other events that by its size or popularity can influence the traffic in the city. These events are obtained from an external

data source (a city event service) which classifies them into the external data group. Road events are dynamic data types that include street works, accidents and any other traffic-related event. STREETLIFE builds a crowdsourcing initiative where users can report these events; therefore the road events are part of the STREETLIFE data.

Maps and cartography, points of interest, public transport information and environmental information are categories of open data that is publically available for the services. This data we define as external data.

3. STREETLIFE KNOWLEDGE REPRESENTATION

This chapter presents the first phase of WP3 work towards STREETLIFE scalable knowledge representation.

We first elaborate on the main STREETLIFE data management architecture, as a subset of the blueprint architecture provided by WP2. Different components that allow employing the data stored in the STREETLIFE data model and integrating the data from other domains are presented in the first subsection.

Since the mobility requirements at the involved pilot sites are different from each other, the implementation of the data management subsystems will be customised according to the local needs. In separate subsections, this chapter describes the data management component implementation at the three pilot sites: Berlin, Rovereto and Tampere.

3.1. Architectural placement

In D2.2.1, the functional view "Management of Mobility Data" of the blueprint architecture identifies the import of data, data validation, data aggregation, subscription, event analysis and the data source broker as main functional components for mobility data management. The rest of the data is also covered through the "STREETLIFE App functionality" and the "Mobility Management and Emission Control" functional views. Another aspect of the data management is covered by the information flow discussion (D2.2.1, Section 6.4). In this section we present a general mobility management architecture that covers the functionalities depicted in the blueprint architecture and defines the main components for data integration and management. We first give a high-level overview of the main components that are going to be part of the STREETLIFE data management system (Figure 3). In the second part of this section we describe the data exchange mechanisms between different components in the system.

Figure 3 presents the high-level data management architecture of the STREETLIFE mobility information system. The system, seen from a data management perspective, has two main groups of components: backend and frontend components.

STREETLIFE backend has two main subgroups: the data management components and STREETLIFE backend services.

Under backend services we define the services that are running on the server side, including the server side of the STREETLIFE application and the Control Panel. Examples server side services are the gamification engine and the route planner. STREETLIFE backend services communicate with the data management components for providing and retrieving STREETLIFE data. External data is integrated through a direct link between the services and the external data providers.

SREETLIFE data management is a collection of components that are part of the STREETLIFE system for integration, interpretation and correlation of STREETLIFE data. *Integrated mobility data management, Real-time data management, Non-real-time data management, Historic data storage, Data correlation and analysis, STREETLIFE Streaming API* and *STREETLIFE Query API* are the components that support the management of the three data categories: real-time, non-real-time and historic data.

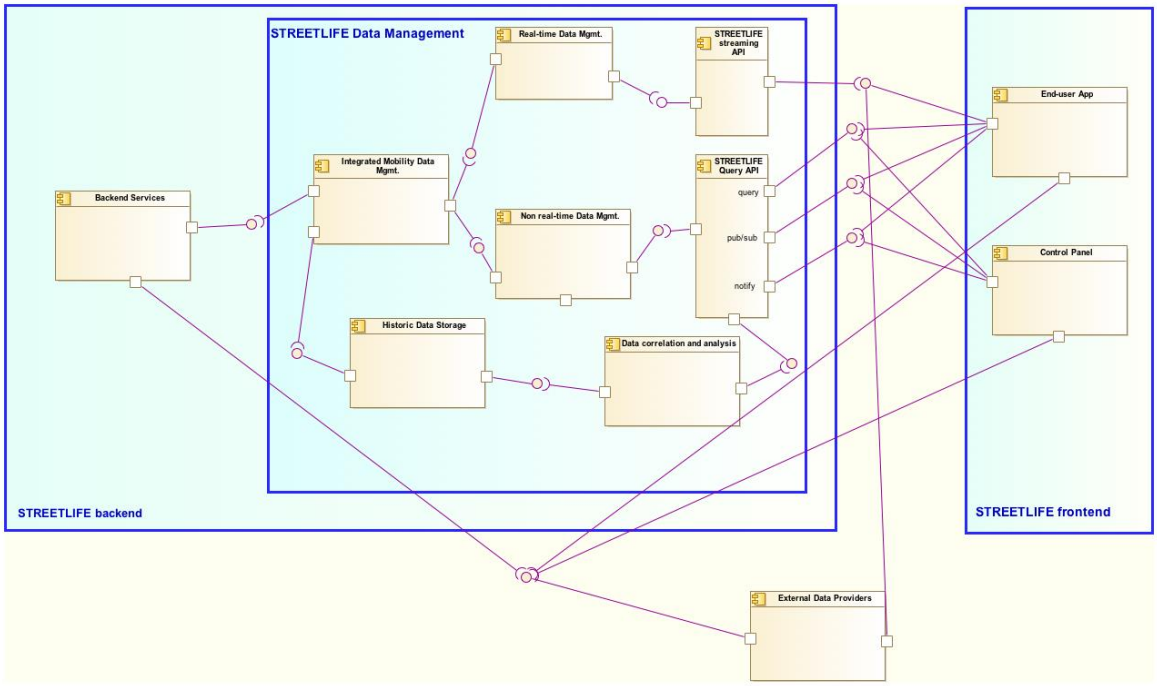

Figure 3: Functional elements from the STREETLIFE data management perspective

Integrated mobility data management's function is seamless integration of different STREETLIFE data sources into the SDM format. Depending on the type of input data, this component communicates with the corresponding data management component.

Real-time data is processed by the *Real-time data management* component and provided by the *STREETLIFE Streaming API* to the consumers of this data.

Non-real-time data is also processed by a parallel pipeline of components, providing integration, management and accessibility for the data by the *Non real-time data management* and *STREETLIFE Query API* components.

The historic data is managed by the *Historic data storage* component. The historic data and the non-real-time data are additionally connected with the data correlation and analysis component responsible for online data processing for improved services and personalization purposes.

The pilot-specific architectural descriptions of the data management components and their implementation are city-specific and differ from each other. In the next section we describe the pilot-specific data management architecture of the three STREETLIFE pilots in Berlin (BER), Tampere (TRE), and Rovereto (ROV) including its relation to the here presented high-level architecture.

To discuss data representation and usage in the three pilots, it is especially important to also understand how data circulation occurs in each of them. Data communication in the various instantiations of the STREETLIFE architecture follows a few general patterns that occur across pilots. Below, we describe them in the form of UML sequence diagrams. The focus of

those descriptions is on data exchange patterns across major subsystem of the STREETLIFE mobility information system (as opposed to detailed data exchanges within each subsystem).

From the Mobility Data Integration perspective, we recognize the subsystems and inter-subsystem data flows depicted in Figure 2; we will present the patterns in general usage by means of notable examples of such inter-subsystem data exchanges.

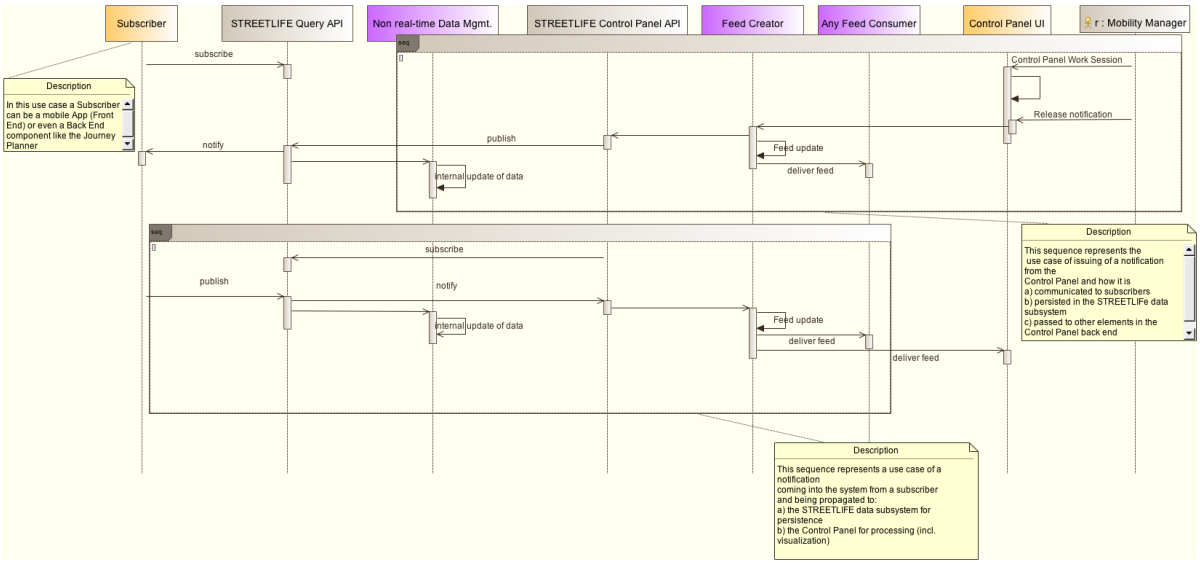

Figure 4: Example of asynchronous data exchange between the STREETLIFE Control Panel and other parts of the system.

In Figure 4, we show a notable example of asynchronous data exchange between STREETLIFE frontend services and backend services; the example focuses on the asynchronous data flows involving the STREETLIFE Control Panel subsystem, which takes equally often the role of data producer and consumer, and also includes some frontend components (like the Control Panel UI) and some backend components.

In the sequence diagram, the STREETLIFE Control Panel UI and the Subscriber are frontend (in orange), whereas backend elements are shown in purple and include not only the STREETLIFE data storage proper (in this particular example, an instance of the *Non Real-Time Data Mgmt.* component of Figure 2), but also the backend components of the Control Panel, like the Feed Creator and various possible additional Feed Consumers besides the Control Panel UI. Other components, shown in grey in Figure 4, represent functional interfaces that inter-connect backend and frontend services; in this example, the Control Panel API and the STREETLIFE Query API are both involved.

The sequence diagram in Figure 4 makes evident how asynchronous data exchange is used pervasively in STREETLIFE: both the publish/subscribe/notify notifications (used across STREETLIFE subsystem) and the feeds (used within the STREETLIFE Control Panel software) are different forms of it.

The second example regards the usage of synchronous data exchanges. This example is shown in Figure 5 and is derived from the sequence diagram about the analysis of historic data included in the design discussion of the STREETLIFE Control Panel subsystem in

D4.2.1[2]; however, here we want to highlight the data exchanges that occur across sub-systems, i.e., between the Control Panel subsystem and the Data Management subsystem.

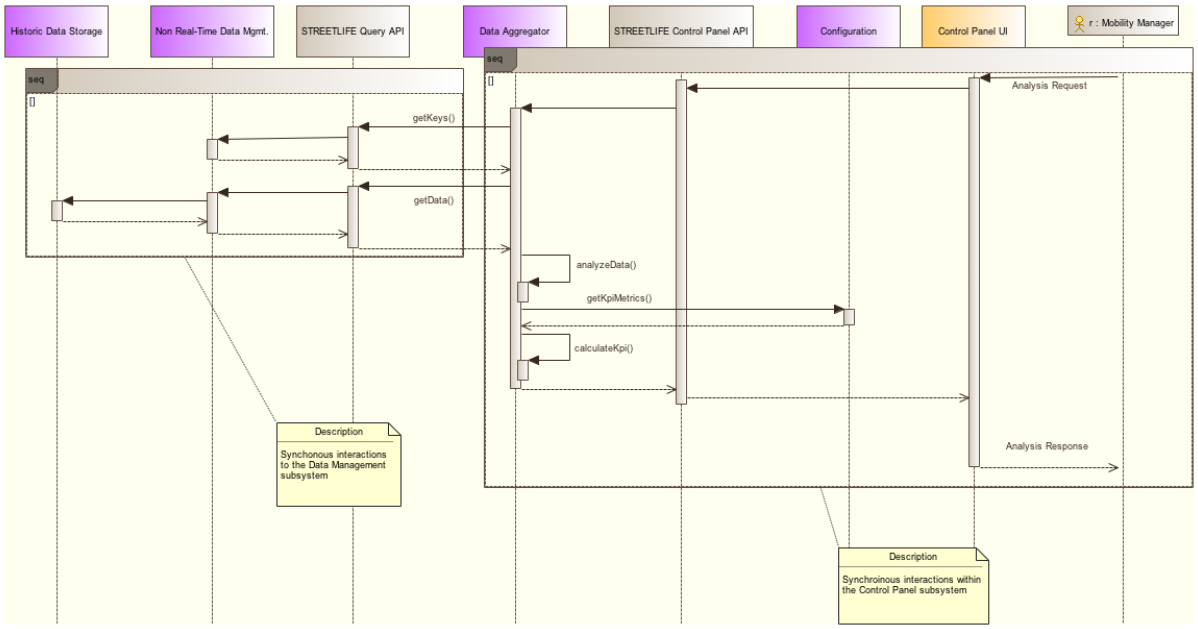

Figure 5: Example of synchronous data exchange between the STREETLIFE Control Panel and the Data Management subsystem.

In particular, Figure 5 shows data exchanges in between backend components of the STREETLIFE Control Panel (*Data Aggregator*) and other backend components taking part in the Data Management subsystem (the *Non Real-Time Data Mgmt.* and the *Historical Data Storage* components). The latter are queried synchronously by the former to retrieve historical data that is needed by the Control Panel, for instance, for the purpose of calculating environmental KPIs of STREETLIFE, and visualizing them in the Control Panel User Interface.

3.2. Pilot Implementation

In this section we present the designs of the data management subsystems at the three pilot sites. The current development of the system depends directly on the use cases the pilots are concentrating on.

3.2.1. ROV

In the ROV pilot, support for the exchange, handling and storing of data belonging to the SDM is implemented by means of a combination of technologies.

Data that is produced by - and exchanged with - other parts of the STREETLIFE mobility information system (in particular by applications that are developed within WP5 and the mobility management control panel developed within WP4) are managed largely by means of a set of integrated FIWARE components. FIWARE [6] is a program within the larger EU PPP Future Internet initiative [7]. FIWARE aims at developing open middleware for the Future Internet and its various emerging application domains, including SmartCities, which makes

available as a set of inter-operable technologies, called “enablers” for experimentation, and to foster innovation throughout the EU ICT eco-system.

It is noticeable that, since those FIWARE enablers are supposed to remain of general use across many domains and application, and to foster inter-operability of possibly very diverse data sources over the Internet, its specifications place very little demands on the type and format of data that it handles, and makes no assumption on the data semantics and models to be enforced. As such, the representation of data flowing through the FIWARE components is extremely general, and only requires adopting contemporary syntactic standards for Internet data, such as JSON. Therefore, while on the one hand, the FIWARE subsystem supports easy inter-operation and integration of different technologies around itself, on the other hand it cannot enforce compliance with the semantics of a data model such as the SDM.

For that reason, compliance with the SDM in the ROV pilot is ensured by means of another processing layer, implemented via KAPcom, a technology developed by the STREETLIFE DFKI partner [8], which we use to aggregate, re-format or otherwise transform raw data items exchanged and stored by means of the FIWARE subsystem into the corresponding elements of the SDM; we then store these more refined data items again within the FIWARE subsystem, thus enabling their fruition to all parts of STREETLIFE through the same uniform data exchange means. This two-layer processing scheme allows us to preserve the generality, open interoperability, and ease of integration offered by FIWARE technology, and couple it with the needs of the STREELIFE projects to represent and reason about the mobility data it produces and handle according to common and higher-level semantics, which provided by the SDM. Moreover, the KAPcom system lends itself also to be used for data correlation and thus enables that kind of analysis, as discussed in Section 7.

We regard the integration of FIWARE software and KAPcom as a **research-oriented prototype**, which is a by-product of collaboration between STREETLIFE partners FBK and DFKI across WP3 and WP6. We intend to incrementally develop this research prototype – and validate it within the ROV pilot - in order to satisfy the data representation and analysis requirements, as well as the architectural constraints, of STREETLIFE. The output of this activity will be a tool that is Future Internet-ready, and will represent an example of Future Internet technologies applied to Smart Mobility. The architectural placement of this research-oriented prototype – in terms of the Component diagram shown in Figure 4 – is that it can take the role of the *Non Real-Time Data Mgmt* component (which is in practice functionally equivalent to *the Integrated Mobility Data Mgmt* component for the first iteration of the ROV pilot, since we do not plan to accommodate real-time information streams in the ROV pilot architecture at this stage).

Besides the two-layers data processing facility represented by the combination of FIWARE software and KAPcom, the ROV pilot also makes use of a number of *legacy data components.* These are data repositories that provide service to pre-existing software elements that take part in the ROV pilot; they include, for instance, any open data sources, or derived open services that wrap such sources and that are used by existing mobility services and applications that are in use in the city of Rovereto and by its citizens, many of which have been detailed in D3.1[1]. Examples include some of the data sources, such as the routes and schedules of public transportation, which are consumed by the journey planner at the core of the ROV mobility App “ViaggiaRovereto”, as well as the maps and Points of Interest (POIs) overlaid on them. Another notable example is the OpenStreetMap data [9] that is used by the

journey planner and mobile apps in ROV as a standard source of mapping and geo-location information and rendering

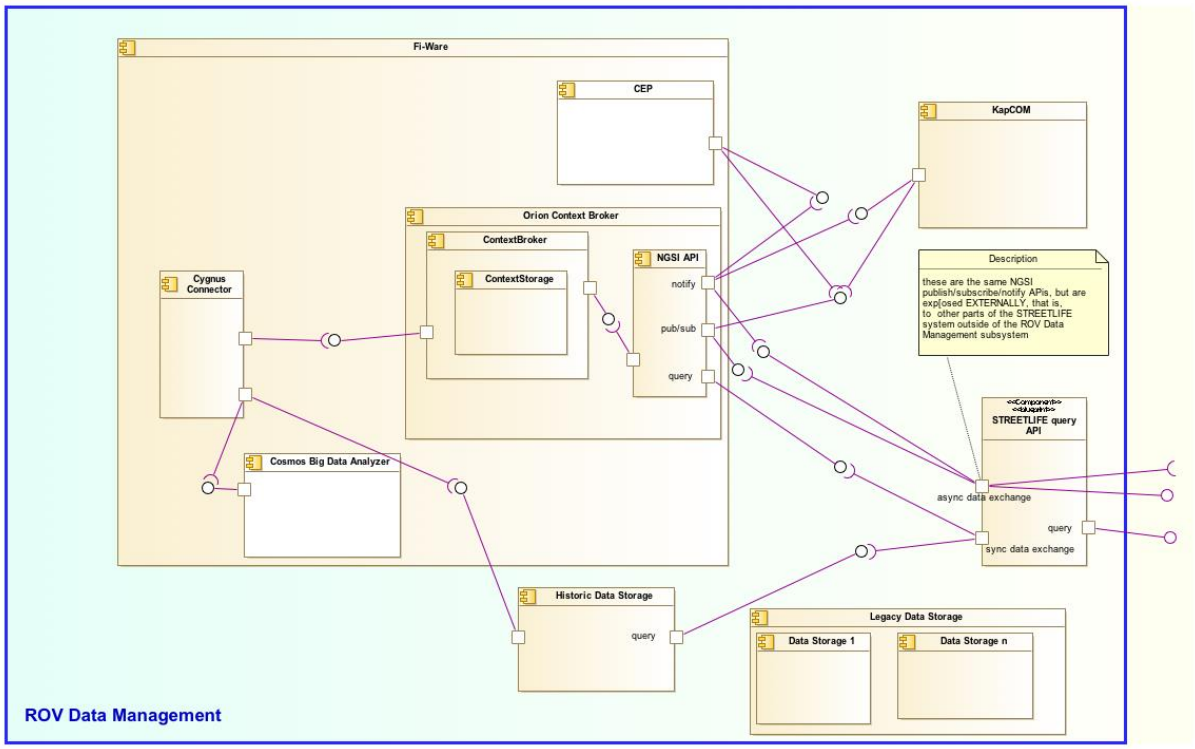

Figure 6: Design of the data management subsystem of the ROV pilot.

Figure 6 depicts the design of the data management subsystem for the ROV pilot. A detailed look at the diagram shows two recognizable macro-blocks. We have already discussed the set of *legacy data storage components*; they are shown as an isolated block in the component diagram in Figure 3, since these data sources are mostly integrated ad hoc and in a point-to-point fashion with the specific components of the ROV pilot architecture that need them, and do not in general contribute to the SDM.

The other significant macro-block visible in the same component diagram is the FIWARE subsystem. In the remainder, we provide detail about the internal structure of this subsystem and the responsibilities of its various elements.

The central component is the FIWARE *Orion Context Broker* [10]. The responsibility of the Context Broker is to store and make available current mobility information that is produced, consumed or exchanged by other parts of the STREETLIFE mobility information system. The information may represent, for example, the current state of a road, or of an urban transportation system. The information is stored as presented by the component producing it, typically according to a JSON format in use by the producer.

The Orion Context Broker offers comprehensive programmatic interfaces (APIs) for writing and reading information, which comply with the OMA NGSI specifications for context data exchange [11]. In particular the NGSI APIs offered by Orion cover OMA NGSI-9 and NGSI-10 API specifications, and enable the following modes of interaction:

- a data producer component can register with the broker meta-data about the type of context data it is going to produce and pass to the broker, so that the broker can catalogue that context type; it can also update the meta-data to inform the broker about format

- a data producer can write a new piece of context data into the broker, or update an existing piece of context data that is already stored in the broker;

- a data consumer can synchronously query the broker about specific context data;

- a data consumer can synchronously query the broker about the availability of specific types of context data in its catalogue;

- a data consumer can declare its interest about some context information (and updates to it) using a subscription mechanism that predicate on information content; following such subscription, the consumer receives asynchronously that information via a notification, every time its subscription matches the context data stored or updated in the broker;

- a data consumer can declare interest about some specific type of context data using a subscription mechanism that predicates on information meta-data; following such subscription, the consumer is informed asynchronously via a notification, whenever context types matching its subscription are registered;

- a data consumer can update its subscriptions, either about context data content or context data types.

In our component diagram we have highlighted the APIs that comply with NGSI, which are implemented and exposed as REST services as part of the Orion Context Broker FIWARE; in terms of the STREETLIFE architecture, they match the *STREETLIFE Query API* connector, which is also found in Deliverable D2.2.1 as part of the discussion of the ROV pilot architecture. The STREETLIFE Query API is an architectural connector that represents the gateway to/from the data management subsystem of STREETLIFE, and is an instantiation of the higher-level abstraction called *Data End Point* that is found in the STREETLIFE blueprint architecture (also described in D2.2.1). The NGSI APIs offered by the context broker can be seen as a reification of the STREETLIFE Query API architectural connector, and match well the composite nature of that connector, since they support both asynchronous (notifications publish/subscribe style) and synchronous (queries request/response style) data communication.

The ROV data subsystem incorporated another FIWARE element, that is, the *Cygnus Connector*. The role of this element is to enable the logging and persistence of context data items that become out-of-date into an external *Historical Data Storage*, which is distinct from the *Context Storage* component used by Orion to persist context data that is still current. Cygnus has been developed within FIWARE based on the open source technology of Apache Flume [12]; it can be configured to monitor certain conditions according to which context data items in Orion are to be considered not current anymore (for example, because of updates to a context item, or expiration of a timeout) and to transfer them into the historical data storage. In STREETLIFE, this functionality is extremely useful, not only for logging, which was the original purpose for Flume, but also to enable analytics on historical mobility data.

For that reason, and to enable queries to it from other parts of STREETLIFE, the Historical Data Storage is connected to STREETLIFE Query API connector.

The Cygnus Connector also links the Orion Context Broker to the *Cosmos Big Data Analyser* [13]. Cosmos is another FIWARE enabler that provides Map/Reduce [14] processing functionality out of the box, by deploying and wrapping an Apache Hadoop [15] installation for integration with other FIWARE software. Cosmos, therefore, can directly provide a wealth of data analytics capabilities, which become especially valuable as the amount of data collected by the STREETLIFE system grows over time. As such, it can be used as an analytics service from, e.g., the STREETLIFE Control Panel. At the same time, it also offers internally the ability of storing all of that historical data, by means of a Hadoop Distributed File System (HDFS) [16] instantiation, therefore it can take the role of the Historical Data Storage. We have chosen to show the Cosmos Big Data Analyser separately from the Historical Data Storage in Figure 6 for two reasons: 1) to make clear that they play two distinct roles, architecturally speaking, although their implementation may be unified; and 2) because in the ROV pilot - as instantiated for the first project iteration - the Cosmos Big Data Analyzer is not going to be used.

The last FIWARE enabler in Figure 6 is the *Proton Complex Event Processing (CEP)* component [17]. The purpose of Proton is to carry out processing on patterns, sequence and streams of notification flowing into the Orion Context Broker, and to create more complex data items as a result (which can flow back into Orion as higher-abstraction context items). The Proton CEP component focuses especially on the **temporal correlation** of data items, with an emphasis on capturing, and reacting to, patterns that occur in (near) real time. As such, Proton promises to become useful for the implementation of data correlation capabilities in STREETLIFE WP3. The CEP component, which is seamlessly integrated with the Orion Context Broker, thus provides the data management subsystem in the ROV pilot with a viable option for data correlation; however, as for Cosmos, we do not consider using it in the initial iteration of the ROV pilot, and we intend to explore its potential for data correlation during the second iteration. At that time, we will also explore the relationship and inter-operability of the Proton CEP component with KAPcom: we see the temporal correlation functionality offered by Proton as complementary to the correlation capabilities of KAPcom, which are more geared towards the logical inference of new and more complex data.

3.2.2. TRE

General description

The Tampere pilot specific data management builds on the investment the City of Tampere already has made in their Intelligent Transport Systems. The Tampere STREETLIFE pilot will utilize Tampere region passenger information system and journey planner (RTPIS) and enhances it with integration of real-time data feeds.

In the TRE pilot, support for the exchange, handling and storing of data belonging to the SDM is implemented by utilizing Tampere region passenger information system’s memory structures and by means of a combination of technologies.

Figure 7 depicts the design of the data management subsystem in the TRE pilot.

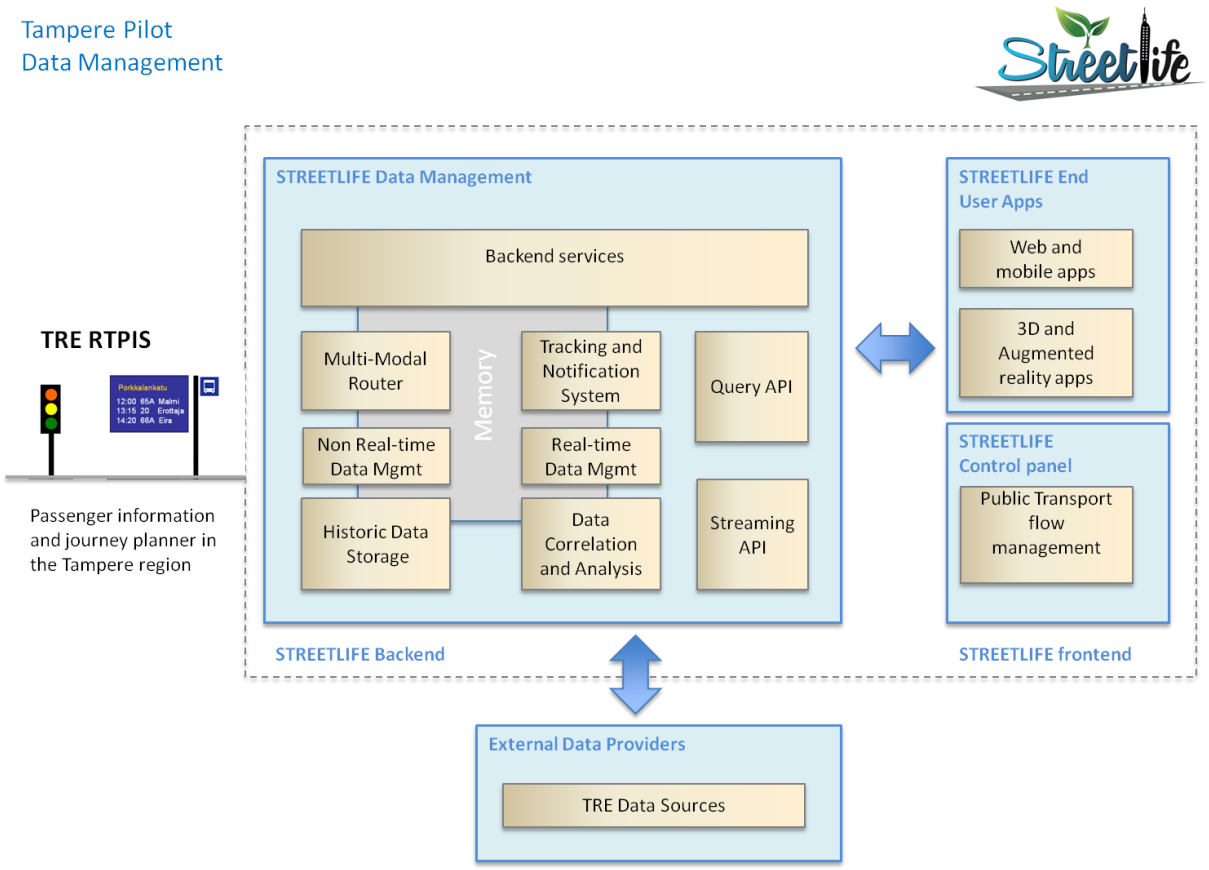

Figure 7: Design of the data management subsystem of the TRE pilot.

Capabilities and Existing Components

The Tampere region passenger information system and journey planner (RTPIS) is capable of handling variety of datasets including real-time data. The solution's key design is that it keeps the relevant data elements in memory structures for fast access. This efficient memory access enables route calculation with user notification based on real-time events. In the Tampere pilot the real-time data integration will fill these memory structures with up-to-date information about bus departure times, the availability of parking places in park and ride locations and crowdsourcing data. With real-time data integration the TRE RTPIS evolves into STREETLIFE pilot where the current system with static data comes to life with real-time data flow. The Tampere pilot Information View specific to Tampere is presented in D2.2.1 Section 11.2.

Real-time public transport data flows in SIRI format. Parking facility data and crowdsourcing data use DATEX II format. In TRE pilot the data is stored and managed in the RTPIS and it resides in the RTPIS memory structures and databases. The data storages are compatible with the STREETLIFE data model. Data consists of static data (i.e. stops, timetables, street names, points of interest, raster maps), selected real-time data elements, user data and statistics as identified earlier in D3.1[1] and D6.1[18].

STREETLIFE extensions

Table 5 is from the STREETLIFE TRE pilot plan and it describes the integration and data correction work related to the TRE data sources in STREETLIFE pilot. The related user scenarios are listed in parentheses.

Table 5: Integration of TRE data sources in STREETLIFE pilot

Data source	STREETLIFE integration/enhancements
TRE-DS1 City road network	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-02, TRE-04 • Adjustment refers to the need to check road network does not have data caps near for example park and ride locations and associated data correction mechanism. • SDM categorization: Static data
TRE-DS2 National road network	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-04) • During the STREETLIFE pilot Tampere public transport region will grow into neighbouring cities. More road network data is integrated. • SDM categorization: Static data
TRE-DS3 Maps of Finland	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-02, TRE-04) • More map data is integrated. • SDM categorization: Static data
TRE-DS4 Street addresses	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-02, TRE-04) • More street address data is integrated.
TRE-DS5 Bicycling network	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-02, TRE-04) • More bicycle network data is integrated. • SDM categorization: Static data
TRE-DS6 Public	• Integrated in RTIPS, adjustment needed (TRE-01,

transport schedules	TRE-02, TRE-03, TRE-04) • More bus and train schedule data is integrated. • SDM categorization: Static data
TRE-DS8 Route Planning API	• Integrated in RTIPS, adjustment needed (TRE-01, TRE-02, TRE-03, TRE-04) • Taking into account all new data • SDM categorization: Dynamic data
TRE-DS9 Real-time bus schedules	• Available, to be integrated (TRE-01, TRE-02, TRE-04) • Data is in SIRI format • SDM categorization: Real-time data
TRE-DS16 Free capacity of parking halls	• Available, to be integrated (TRE-04) • Data is in DATEXII • SDM categorization: Real-time data
TRE-DS24 User profile	• Integrated in RTIPS, adjustment needed (TRE-02, TRE-04) • SDM categorization: Static data with dynamic elements
TRE-DS25 CO2 emissions	• Available, to be integrated (TRE-05) • SDM categorization: Static data

3.2.3. BER

The data management subsystem in Berlin will be implemented as part of the City Intelligence Platform (CIP). CIP encapsulates the components for real-time, non-real-time and historic data management. The data is pre-processed in a way to allow cross domain correlations and stored within a data warehouse. CIP is collecting field data via interconnection to multiple Siemens- and external systems. A main external data source in the BER pilot is the Mobility management centre – VMZ. Analytics and decision support modules can directly access this storage and derive recommendations for a more efficient operation.

Figure 8 shows the main relations of CIP with the frontend components and external data sources. With respect to the Berlin pilot, CIP is used to deliver data to

1. STREETLIFE Mobility Management and Emission Control Panel
2. STREETLIFE Mobile App Service

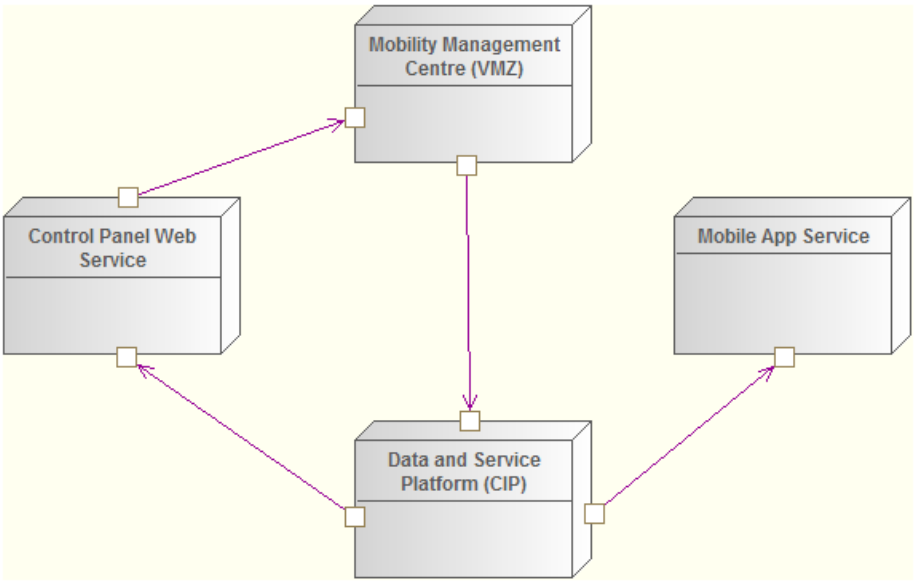

Figure 8 Design of the data management subsystem of the BER pilot

From an architectural point of view, CIP supports two main paradigms to deliver data to aforementioned consumers:

- Request/response to get data from REST services
- Pub/sub to receive events or data asynchronously

Current Berlin setup

CIP is used to support STREETLIFE use cases in the context of STREETLIFE Control Panel and KPI calculation (WP4) and mobility apps (WP5).

1. Concerning KPI calculation and dashboards, CIP uses an Export, Transform, Load (ETL) approach to load bike-sharing data from the Siemens Integrated Mobility Platform (IMP) into CIP. This data depicts the availability of shared bikes by bike sharing station. This information is saved in a relational database and made available for analytics, visualisation and KPI calculation. The analytics module uses the data to calculate statistics and to make demand forecasts.
2. With respect to mobility apps, CIP offers interfaces for:

a. Intermodal route (IMR) planning, which is used by the Berlin Mobility App. The CIP IMR interface uses the IMP IMR router to request trip proposals.

b. In addition, CIP offers a user service, which is also used by the Berlin Mobility App. The user service is used to save the set of trip proposal requested and selected by a particular user.

As described in D2.2.1 there is additional data information defined, to complete STREETLIFE Control Panel computation and visualisation as well as the Mobile App routing and gamification services. The data is stored in CIP data platform or used directly via REST services.

The following two points describe the data flow in CIP

1. The data flow in KPI calculation/dashboards:

    a. CIP imports data via ETL from the Siemens IMP.

    b. CIP saves the data in a relational database.

    c. The computational module requests the data from the database and calculates relevant KPIs.

    d. The STREETLIFE Dashboards (WP4 STREETLIFE Control Panel) request the KPIs from REST services and visualises the KPIs.

2. The data flow concerning the mobility app:

    a. With respect to trip proposals:

        i. The Berlin Mobility App requests a trip proposal specifying a set of criteria.

        ii. CIP receives the request and forwards the request to Siemens IMP.

        iii. CIP receives the response from Siemens IMP and forwards the response to the Berlin Mobility App.

        iv. The Berlin Mobility App request trip proposal details.

        v. CIP receives the request and forwards the request to Siemens IMP.

        vi. CIP receives the response from Siemens IMP and forwards the response to the Berlin Mobility App.

    b. With respect to the user service

        i. The Berlin Mobility App updates the trip history.

        ii. CIP saves the user’s trip history in the database.

In general, CIP supports REST-based services for creating, updating and retrieving data. The data format is JSON. CIP supports HTTP GET operations to make KPIs available to the STREETLIFE Control Panel. CIP supports HTTP POST to request trip proposals and HTTP GET to get proposal details in the context of the STREETLIFE Mobility App. In addition to the trip data, various HTTP operations are provided for the user service support (e.g., updating and saving user trip history).

4. CROWDSOURCING

> ***Crowdsourcing*** *is the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers*[1]

Crowdsourcing data and initiatives can be categorized in different ways. We have found the following two orthogonal dimensions particularly useful to describe and reason about crowdsourcing in the domain of urban mobility data in general, and for STREETLIFE in particular:

- *Static vs. Dynamic crowdsourcing*: these categories refer to whether the crowdsourced data is accumulated in the background and "offline" with respect to the time scale of operation of the system using that data, or is instead contributed "online", and thus constitutes a change to the knowledge and information base that is relevant to the current operation and state of the running system.

- *Active vs. Passive crowdsourcing*: these categories refer to whether the crowdsourcing data is contributed as a result of an explicit action, with which the user produces a piece of information and issues it to the system, as opposed to information that is produced without the user's direct intervention, simply as a by-product of her ICT-mediated activity [19] (like e.g., in *mobile crowd sensing* [20]).

Other categorizations are possible; for example, Static and Active crowdsourcing could be *Moderated* (that is, curated) or *Non-moderated*, which speaks to the trustworthiness and reliability of the final crowdsourced information that is accepted into the system [21].

For the purposes of the project, the Static vs. Dynamic and Active vs. Passive dimensions are sufficient to classify approaches to crowdsourced data and opportunities for its integration in the STREETLIFE mobility information system; Figure 9 shows the quadrants formed by the orthogonal dimensions mentioned above. Figure 9 also provides examples of crowdsourced data for urban mobility that pertains to those quadrants, which are supported in STREETLIFE, and are used – or are planned to be used – in its pilots. Notice that, although several applications of Passive and Static crowdsourcing do exist, for instance in the field of geo-localized passive collection of information (with case studies e.g., on monitoring and mapping environmental conditions and pollutions [22] [23]), there are, as far as we know, no practical use cases for Passive and Static crowdsourcing in the domain of urban mobility

[1] "Crowdsourcing" – Merriam-Webster.com, retrieved 29.8.2014

proper, nor in STREETLIFE specifically. Therefore, the corresponding quadrant of Figure 9 is empty.

An important crowdsourced asset for STREETLIFE is OpenStreetMap [9], which already provides the basis for the STREETLIFE intermodal and personalized travel assistance systems in ROV and TRE, as well as other STREETLIFE services and apps – for details see Deliverable D5.2.1 [24]. OpenStreetMap (OSM) is possibly the single most notable single Static and Active crowdsourcing initiative in the domain of mobility, and represents a critical enabler for mobility and transportation service development worldwide.

	Static	Dynamic
Passive	✖	Travellers tracking (BER)
Active	Open Street Map (ROV, TRE)	Users' notifications (BER, ROV, TRE) Traffic aides' notifications (ROV)

Figure 9: Crowdsourcing classification and use in STREETLIFE.

In the Passive and Dynamic crowdsourcing category, a typical use case in the domain of mobility is represented by the tracking of travellers as they go about their itinerary, for example using GPS tracks, possibly enriched with other information coming from additional sources like public transportation systems, parking systems, ticket validation systems and so on. Travellers tracking is information that can be used for both personalized guidance during one's itinerary (when processed individually), or for deriving corollary information about the areas covered by the itineraries of the users' population (for example current average vehicular speed on a given road, as demonstrated, among others, by the popular Waze App [25]). In STREETLIFE, Passive and Dynamic crowdsourcing is planned to be used for extending the crowdsourced traffic information to different modalities, especially the bicycles. BER pilot together with DFKI will experiment on extracting knowledge from GPS data collected from bicycles in order to provide enhanced cycling map of the city. Passive and Dynamic crowdsourcing will also be used for personalization purposes.

The Active and Dynamic crowdsourcing category covers a multitude of approaches, in which users on the ground are asked to send snippets of information to an information system. Crowdsourced data is then compiled by that system, typically to enrich a pre-existing knowledge base with extra information, which may represent for example current state changes on information entities already represented in the knowledge base. As such, Active

and Dynamic crowdsourcing is a form of citizen participation, and as such, is at times encouraged and rewarded via incentive schemes [26] [27].

In STREETLIFE, Active and Dynamic crowdsourcing scenarios in urban mobility are enabled by *ad hoc* features in several of the mobile apps that are being developed, and which are described in detail in Deliverable D5.2.1.[24] Whereas some apps are geared towards end-users/citizens in general, at least one STREETLIFE mobile app is specifically dedicated to traffic operators. This алтернативе approach, which is somewhat similar to moderated crowdsourcing, is intended to lead to high-reliability data, and high accuracy in the estimation of parking time options and congestion by the ROV instantiation of the STREETLIFE mobility information system.

In the rest of this section we report in further detail the usage of the various types of crowdsourcing discussed above, focusing on what data they generate and how that data is used within the STREETLIFE mobility information system.

4.1. Active and Static Crowdsourcing

As an example for Active and Static crowdsourcing we present in this section OpenStreetMap. OSM is an appraised case of crowdsourcing for collecting spatial data. Total openness, full availability of data and tools have lured a huge community to work on it. With OSM, everyone can produce their own version of the map using its raw coordinate and metadata information in a downloadable database format. The project is maintained by over 15000 active users and anyone can register for free to start modifying the map. Tools are included to view map information on top of Microsoft's satellite images and tracks recorded by users traveling with a GPS receiver. They allow drawing roads, buildings and other features missing in the map database that can be seen in other imagery. Road names and classification can be added based on personally surveying the physical location or from other open data sources. Some governments have contributed their own data to the project.

OSM provides a primary spatial data model and data set for some of STREETLIFE's applications, including the routing algorithms of AALTO and FBK.

4.1.1. OSM structure

The OSM database consists of *nodes*, *ways* and *relations*, all of which can be labelled by associating them with any number of *tags*. A node is simply a pair of coordinates with a unique identifier. A way is an ordered list of nodes and is used to define any complete or partial physical feature that is formed by connecting all consecutive nodes together. Finally, a relation is an ordered list of nodes and ways that are somehow related to each other, each with a specific role which can be any text. This structure is illustrated in Figure 10 (upper).

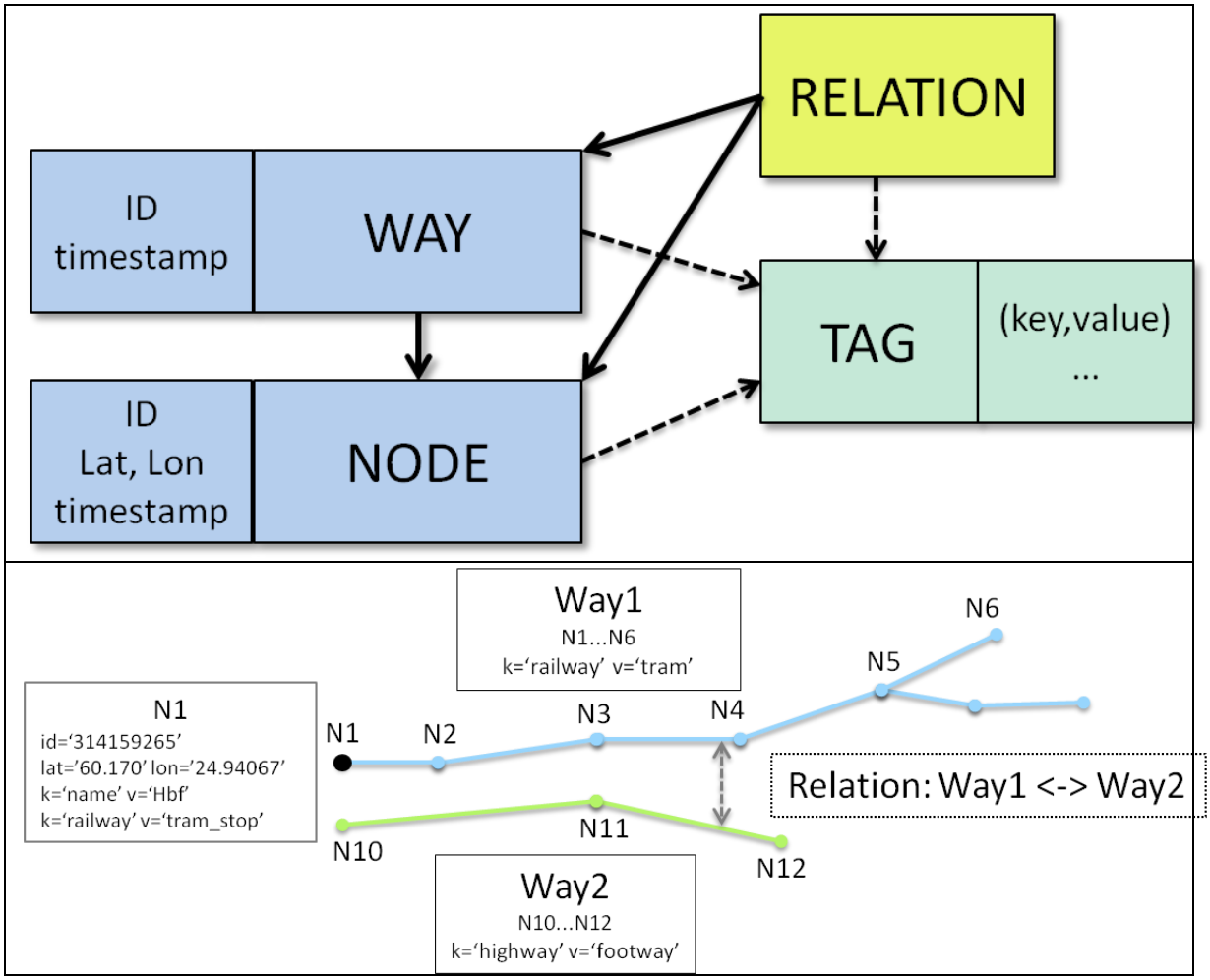

Figure 10. OpenStreetMap elements and an example.

A *tag* in OpenStreetMap is a *key* with an associated *value*. Both can be any text or number allowing a wide variety of data to be represented. For example traffic lights are nodes with a tag using the key ”highway” and value ”traffic signals”. Roads are ways tagged with the key ”highway” and one of many possible values for different road classifications. There’s an entire ecosystem of web forums and collaboratively written web pages to administer the database in a democratic process where everyone has equal power. A process has been defined to suggest new tags, discuss them and vote for a general acceptance.

Most relevantly to routing, public transit lines are defined as relations containing the roads the vehicle travels and the nodes where it will stop to drop off and pick up passengers.

The database is the core of the OSM project and contributors are encouraged to focus on representing data in a consistent way as it appears in the real world instead of trying to make the map look more pleasant. It is then up to other tools to automatically produce graphical maps. Here OSM follows the nearly 2000 year old idea of separating data from its visualization [28].

4.1.2. OSM availability and formats

OSM data is released under the Open Database License (ODbL). The data is available in two common formats, XML and PBF. The XML file is purely text and the commands to define nodes, ways and relations are English words so it guarantees broad understanding of the structure of the file. As an example, a tram stop and its platform extracted from the public OpenStreetMap database are shown in Table 6. The node represents where passengers can get in and out of trams, and its number 314048973 is referenced from the platform and several relations describing tram lines. Three other nodes of the platform are shared by footways connected to nearby sidewalks. Thus there's a topological connection between the pedestrian road network and public transit network, and connections can be found by simply examining node numbers.

Unfortunately the XML representation is not very compact. The text-based database extracts are being replaced by PBF, Google's Protocolbuffer Binary Format [29]. PBF allows describing structures with fields of various types, including lists of other structures. In OpenStreetMap the database is stored in a PBF package divided into a list of blocks which are individually compressed using DEFLATE [30]. In effect, a map database of 3.3GB XML data can be expressed with only 159MB (the case of Finland), together with faster process times.

Table 6. Example data downloaded from the OpenStreetMap database

```
<node id='314048973'
    timestamp='2013-01-02T17:14:17Z'
    uid='4660' user='alv' visible='true'
    version='12' changeset='14502699'
    lat='60.170435' lon='24.9406728'>
  <tag k='departures_board' v='realtime' />
  <tag k='name' v='Rautatieasema' />
  <tag k='name:fi' v='Rautatieasema' />
  <tag k='name:sv' v='Jrnvgsstationen' />
  <tag k='railway' v='tram_stop' />
  <tag k='ref' v='0302' />
  <tag k='shelter' v='yes' />
  <tag k='wheelchair' v='yes' />
</node>
```

```
<way id='27265250'
    timestamp='2012-07-29T04:24:34Z'
    uid='38239' user='Daeron'
    visible='true'
    version='7' changeset='12530429'>
  <nd ref='299269501' />
  <nd ref='299269500' />
  <nd ref='314048973' />
  <nd ref='299269003' />
  <nd ref='1003278949' />
  <tag k='highway' v='platform' />
  <tag k='lit' v='yes' />
  <tag k='surface' v='paving_stones'/>
</way>
```

4.1.3. OSM Integration

The structure of OSM provides a very flexible and natural basis for creating rich map-based data sets that can be enhanced in multiple ways and can be modularly enriched. For the purposes of STREETLIFE, this enrichment process is used to incorporate other geo-located data, such as public transportation data, e.g. in the GTFS format elaborated in Section 6.1, and obtain: a) the primary data set used for journey planning; b) app-specific maps that superimpose additional data on the regular urban map; and c) a World State skeleton for real-time data management, which is exemplified in AALTO's real-time data pipeline described in Section 3.2.2 above.

Although with different implementation details, the same basic enrichment process is followed by STREETLIFE partners AALTO and FBK in their software, and for the TRE and ROV STREETLIFE implementations. Figure 18 presents the enrichment process step by step. An OSM data set covering a region (such as a city map) is downloaded and relevant roads along with their tags extracted. Public transit stops and routes are fetched, for example by a GTFS data set, and extracted. Other data sources can be similarly integrated, as deemed useful, such as parking locations, traffic light locations, bike sharing stations, etc. All of those locations are matched to nearest street and overlaid onto the basic map topology. This enriched set is then stored aside in OSM format for review with any OSM tools. Finally, (and optionally, for example this is not used in the ROV case) the map can compressed and encoded for efficient storage, transmission and utilization. The implementation of this process has been for example encapsulated by AALTO in its Open Source *OSM Squeezer* tool.

4.2. Active and Dynamic Crowdsourcing

Active and Dynamic crowdsourcing has already been incorporated in a variety of STREETLIFE mobile apps.

For example, the mobile apps developed by FBK and being deployed in the first iteration of the ROV pilot, i.e. the extended ViaggiaRovereto App, the Bike Sharing App, and the Parking Availability App, which are described in detail in D5.2.1[24], all include crowdsourcing features.

In ViaggiaRovereto, which is a pre-existing personal mobility assistance app developed by FBK and available from the Google Play App Store with an already an active user base in ROV. This app is being extended and enhanced with respect to STREETLIFE requirements and the use cases of the ROV pilot, Active and Dynamic crowdsourcing features allow citizens to contribute information about public transportation delays of urban or long-range buses and trains. These delay notifications are issued by the app in JSON format; a typical crowdsourced notification from a ViaggiaRovereto is shown on Figure 11:

```
POST /[aaa]/[bbb]/[ccc] HTTPS/1.1
Host: [xxx].smartcampuslab.it
Accept: application/json
Authorization: Bearer {user access token}

{
   "position":{
      "lon":"11.100248",
      "stopId":{
         "id":"22500c",
         "agencyId":"12"
      },
      "name":"RONCAFORT nord",
      "stopCode":"22500c",
      "lat":"46.103456"
   },
   "id":"71d6a009-dec9-4e1b-bd3b-d69ca65826e4",
   "to":1377094660792,
   "effect":"effect",
```

```
   "transport":{
      "tripId":"01-Feriale_036",
      "routeShortName":"01",
      "routeId":"01",
      "type":"BUS",
      "agencyId":"12"
   },
   "description":"description",
   "creatorId":"1",
   "delay":300000,
   "creatorType":"USER",
   "from":1377094360792,
   "type":"DELAY",
   "note":"note"
}
```

Figure 11: Example of crowdsourced information - public transportation delay.

In the ROV Bike Sharing App, an end user can leverage the Active and Dynamic crowdsourcing features to send a notification to the system. The notifications can be about any problems related to the vehicle she has been riding, or a bike station, or even to express her general feedback about the bike sharing service. These event notifications are issued by the app in JSON format, and may include a digital image of the problem being notified; a typical crowdsourced notification from a user of the Bike Sharing App is shown in Figure 12:

```
POST /[aaa]/[bbb]/[ccc] HTTPS/1.1
Host: [xxx].smartcampuslab.it
Accept: application/json
Authorization: Bearer {user access token}

{
  "date": 1408574559678,
  "cityId": "rovereto",
  "objectType":"Station", // type of notification
  "objectId": "12345",
  "report": "bla-bla", // user-provided feedback
   "fileId": "1231321231" // identifier of picture file provided by the user.
}
```

Figure 12: Example of crowdsourced information – bike sharing feedback.

The ROV Parking Availability App is different from the cases of Active and Dynamic crowdsourcing described above, since that app is geared towards traffic aides working in the city. It enables them to report on parking availability on the streets and in certain parking lots and facilities in the city of Rovereto, which are not equipped with an infrastructure for real-time counting capabilities for residual capacity. Because of its users, the parking availability data obtained via this app is analogous to curated crowdsourced data. The app will be used in the first iteration of the ROV pilot to collect data about occupation of parking slots in critical areas of the city, which initially is used for offline statistical analysis for the ROV mobility

manager, Later on, this curated crowdsourced information will be leveraged to adapt in real time the travel recommendations to end users who mean to drive their car to over-crowded areas of the city. The parking availability notifications are issued by the app in JSON format; a typical notification from a traffic aide is shown in Figure 13:

```
POST /[aaa]/[bbb]/[ccc] HTTPS/1.1
Host: [xxx].smartcampuslab.it
Accept: application/json
Authorization: Bearer {user access token}

{
  "_id" : "street@rovereto@51489136975af42ddd711127",
  "slotsFree" : 10,
  "slotsOccupiedOnFree" : 10,
  "slotsUnavailable" : 1,
  "slotsPaying" : 10,
  "slotsOccupiedOnPaying" : 1,
  "slotsTimed" : 9,
  "slotsOccupiedOnTimed" : 9,
  "polyline" : "webwG{~kbAD?F?\\@X@t@@",
  "agency" : "rovereto",
  "position" : [
                45.89164635632249,
                11.043825298547745
               ],
  "lastChange" : {
                  "author" : "author",
                  "time" : 1408574559678
                },
  "name" : "Corso Bettini ",
}
```

Figure 13: Example of curated crowdsourced information – parking availability on a city street.

4.3. Passive and Dynamic Crowdsourcing

Floating data collections have been already used for obtaining travel time and speed data from vehicles and present essential source for real-time traffic information. The process is based on the collection of localization data, speed, travel direction and time information from mobile phones, GPS devices or other sensors available in vehicles. In STREETLIFE we propose to use floating bicycle data as a source for defining cycling routes and cyclists' behavior. The fleet presents a group of bicycles that are being used continuously and by variety of different users. Therefore, they present a valuable source of diverse traveling information that covers larger area of the city. Perceiving the advantages, GPS data from bicycle fleets will be considered first as main floating bicycle data source for this project.

Taking in consideration that cyclists belong to a vulnerable group of people that is influenced by many external factors like weather, topography, personal shape etc., the problem of route definition for cyclists extends to the problem of: Which routes are suitable for specific type of users and current set of conditions? Additional to the electronic traces from GPS we will

consider external data sources relevant to cyclists. The goal is to define a cycling route network beyond what existing maps and routing services provide. Correlating different data sources will create a complex cycling profile model that will be then used for route recommendation and personalization.

Figure 14 illustrates collection of bicycle floating data as an example for Passive and Dynamic crowdsourcing. As shown on the figure, the core crowdsourcing component receives an input from three different components responsible for past data from taken routes, real-time floating data from cyclists and external relevant data (additional sensors, events, traffic etc.).

The core component will incorporate techniques for data correlation, map generation, user classification and real-time route generation that together contribute to enhanced cycling route network generation. As an individual, every user feeds data to the real-time component where user profiling and personalization services are provided. As a part of the fleet every user contributes in creating the global image of the bicycle traffic situation as well as in traffic event discovery. The aim is to develop general learning and modeling techniques that will be applicable for any urban area.

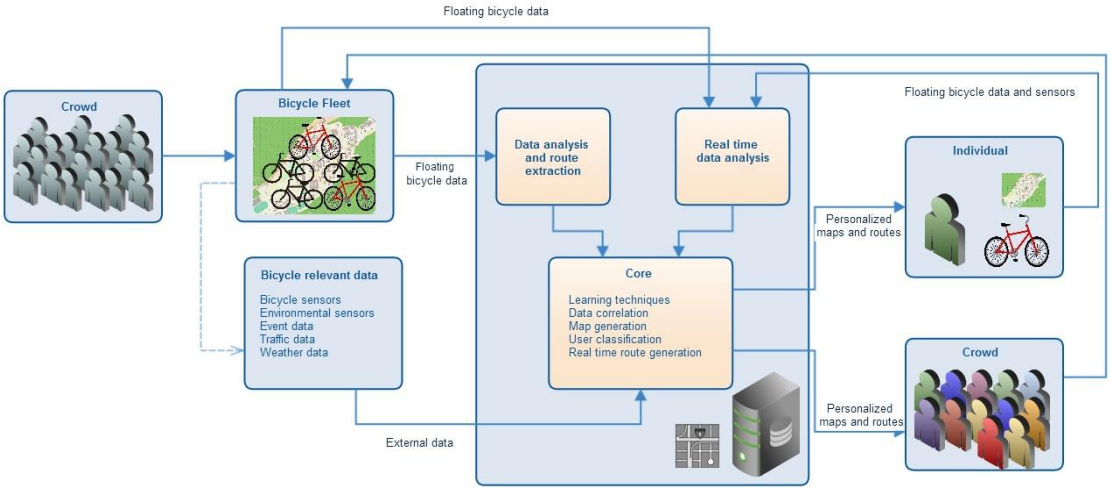

Figure 14: Example of Passive and Dynamic crowdsourcing by collection of bicycle floating data

STREETLIFE partner DFKI owns a small bicycle fleet at the campus of Saarland University. The solution developed in WP3 will be initially tested with that bike fleet. The plan for further testing in Berlin will be additionally developed.

5. REAL-TIME DATA

STREETLIFE utilizes static, dynamic and real-time data for the various applications. The definition of "real time" varies greatly between domains. In computer science, with the context of operating systems, the design goal is a guarantee of performance, minimizing any delays and jitter in task execution times. Processing time requirements are in the order of tenths of seconds or shorter. In other fields, the concept becomes more vague and related variables less strict. Especially when system performance is measured by human perception, the requirements become highly task dependent. In public transportation, for example, a passenger information display system (implemented at bus stops with monitors or as mobile apps) is considered to be a real-time application even if the granularity is in the order of minutes.

We plan to verify our requirements and our application performances perceptually for those tasks where latency is critical in later stages of the project. We expect the requirement to be most significant for the mixed reality applications of WP5.

Integration of real-time data in STREETLIFE requires, in a common case, association with underlying static data sets such as street topology and bus routes. For example, crowdsourced floating GPS data can be used to infer the current traffic flow, which again may affect weights of road network segments for routing – and the road network itself can be based on a crowdsourced static data set. Please refer to Section 4 of this document for discussion on crowdsourced data.

5.1. Real-time data sources

STREETLIFE real-time data sources are as multifaceted as are our applications. We generally utilize two types of real-time data sources: *passive* and *active*. Passive sources are *automated measurements*, which provide a raw data feed, possibly requiring further processing to be applicable. These sources include point sensors such as inductive loops, video observations, infrared and laser vehicle detection, street weather stations and floating GPS. Active sources include human interaction for defining each *event.* These include crowdsources where citizens report anomalies in traffic, official reports such as those from municipalities, emergency services and police or traffic administration facilities and public transport operators where specialized personnel create notifications and other messages.

D3.1/AppendixA [1] presents a list of identified external data sources available for STREETLIFE, including real-time data, for each site. Internally, our end-users may act as active or passive data sources via our STREETLIFE applications.

5.2. Real-time tracking: Tampere public transportation

Of the multiple STREETLIFE real-time data sources, we present one specific and well-specified case to illustrate the use of real-time data. The city of Tampere provides a real-time public transportation tracking interface as an Open Data service. This service covers bus locations of Tampere City Transport, provided by Tampere region Real Time Passenger Information System (RTPIS).

5.2.1. Tampere real-time tracking service description

Tampere City Transport's bus fleet tracking is available in the SIRI Lite format [31] encoded in JSON (JavaScript Object Notation). This SIRI feed covers the larger Tampere area with approximately 200 tracked vehicles. The SIRI real-time services are implemented as *request/response* model web services. This means that each request must contain information which service/resource is to be consumed and what are the characteristics of the data that the consumer is expecting. As a response for a *VehicleMonitoringRequest*, for example, the interface provides a single file containing vehicle monitoring responses for all vehicles in circulation. The file changes once per second with an additional latency of one second before location observation updates from GPS receivers in the vehicles are centrally received and written into the file.

The SIRI real-time services must be used only for server-2-server communications. This means that applications (e.g. web applications and mobile applications) must not make direct requests to the real-time SIRI web services.

This limitation is caused by the fact that the infrastructure for Tampere Public Transport Information System cannot carry load from thousands or tens of thousands of consumers. Also, since the SIRI XML format is so verbose and XML processing in client side requires lots of resources, it is recommend that applications using the real-time SIRI data should use other formats such as JSON when communicating between client and backend.

Table 7 provides a list of services available from the Tampere SIRI interface. Update intervals are described in Table 8.

Table 7: Tampere SIRI Services

Service	Definition
Stop Monitoring Service	• The SIRI Stop Monitoring Service provides a stop-centric view of vehicle arrivals and departures at a designated stop. It can be used by displays and other presentation services to provide departure board and other presentations of timetable and real-time journey information both at stops and at a distance.
Vehicle Monitoring Service	• The SIRI Vehicle Monitoring Service reports the position of a vehicle or group of vehicles making monitored journeys in real-time. It can be used to monitor the progress of vehicles, to provide information for systems which present visualisations of the movement of vehicles, for example on maps, lists or line diagrams, and to exchange information about roaming vehicles with other control centres.
General Message Service	• The SIRI General Message service is used to transmit messages between the participants. The data to be published will typically be informative messages such as travel news and other operational advice, entered or

	forwarded into the system, normally by a control centre.
HTTP Basic Authentication	• Tampere SIRI interface uses HTTP Basic Authentication, which is a simple challenge and response mechanism with which a server can request authentication information (a user ID and password) from a client. The client passes the authentication information to the server in an Authorization header. The authentication information is encoded with base-64 encoding.
Capability Services	• StopMonitoringCapabilitiesRequest • VehicleMonitoringCapabilitiesRequest • GeneralMessageCapabilitiesRequest

Table 8: SIRI interface update intervals

Service	Request Interval (seconds)	Background
VehicleMonitoring	1	• Vehicles are positioned once every second.
StopMonitoring	30	• Stop calls are estimated with accuracy of one minute.
GeneralMessaging	60	• The SIRI General Message service is used to transmit messages between the participants. The data to be published will typically be informative messages such as travel news and other operational advice, entered or forwarded into the system, normally by a control centre.

5.2.2. SIRI Examples

Request to get data for all vehicles:

```
POST /ws HTTP/1.1
Content-Type: application/xml;charset=UTF-8
```

```
Authorization: Basic dGVzdDp0EESAZXN0
Content-Length: 649
Host: https://siri.ij2010.tampere.fi
Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3"
xsi:schemaLocation="http://www.kizoom.com/standards/siri/schema/1.3/siri.xsd">
	<ServiceRequest>
		<RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
		<RequestorRef>YOUR_APPLICATION_ID</RequestorRef>
		<VehicleMonitoringRequest version="1.3">
			<RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
			<VehicleMonitoringRef>VEHICLES_ALL</VehicleMonitoringRef>
		</VehicleMonitoringRequest>
	</ServiceRequest>
</Siri>
```

Request to get data for vehicles operating on line 30:

```
POST /ws HTTP/1.1
Content-Type: application/xml;charset=UTF-8
Authorization: Basic dGVzdDp0EESAZXN0
Content-Length: 649
Host: https://siri.ij2010.tampere.fi
Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3"
xsi:schemaLocation="http://www.kizoom.com/standards/siri/schema/1.3/siri.xsd">
	<ServiceRequest>
		<RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
		<RequestorRef>YOUR_APPLICATION_ID</RequestorRef>
		<VehicleMonitoringRequest version="1.3">
			<RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
			<LineRef>30</LineRef>
		</VehicleMonitoringRequest>
	</ServiceRequest>
</Siri>
```

Response:


```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Wed, 24 Apr 2013 06:41:42 GMT

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:ns2="http://www.ifopt.org.uk/acsb"
xmlns:ns3="http://www.ifopt.org.uk/ifopt"
xmlns:ns4="http://datex2.eu/schema/1_0/1_0" version="1.3">
       <ServiceDelivery>
              <ResponseTimestamp>2013-04-24T09:41:42.486+03:00</ResponseTimestamp>
              <ProducerRef>IJ2010</ProducerRef>
              <Status>true</Status>
              <MoreData>false</MoreData>
              <VehicleMonitoringDelivery version="1.3">
                     <ResponseTimestamp>2013-04-
24T09:41:42.486+03:00</ResponseTimestamp>
                     <Status>true</Status>
                     <VehicleActivity>
                            <RecordedAtTime>2013-04-
24T09:41:42.173+03:00</RecordedAtTime>
                            <ValidUntilTime>2013-04-
24T09:42:12.173+03:00</ValidUntilTime>
                            <MonitoredVehicleJourney>
                                   <LineRef>30</LineRef>
                                   <DirectionRef>2</DirectionRef>
                                          <FramedVehicleJourneyRef>
                                          <DataFrameRef>2013-04-24</DataFrameRef>
                            <DatedVehicleJourneyRef>0935</DatedVehicleJourneyRef>
                                   </FramedVehicleJourneyRef>
                                   <OperatorRef/>
                                   <OriginName xml:lang="fi">Keskustori M
</OriginName>
                                   <DestinationName xml:lang="fi">Etelä-
Hervanta</DestinationName>
                                   <Monitored>true</Monitored>
                                   <VehicleLocation>
       <Longitude>23.77120000000000</Longitude>
       <Latitude>61.49190000000000</Latitude>
                                   </VehicleLocation>
                                   <Bearing>148.0</Bearing>
                                   <Delay>P0Y0M0DT0H2M58.000S</Delay>
                                   <VehicleRef>TKL_248</VehicleRef>
                            </MonitoredVehicleJourney>
                     </VehicleActivity>
              </VehicleMonitoringDelivery>
       </ServiceDelivery>
</Siri>
```

5.3. Real-time Events: DATEX II for standardized exchange of traffic information

Primarily, STREETLIFE online crowdsourced data is received by STREETLIFE services in any of our internal protocols. In selected cases where the data involves localized events and the state of the traffic, we also publish this data. To be compliant with external services, this output will be available as a standardized **Datex II** compliant service[2]. For example, in the TRE Pilot, AALTO will receive crowdsourced data as part of their advanced graphical applications, and provide a Datex II feed as output. This Datex II feed is then received by CGI and used for the TRE Pilot.

DATEX II is a CEN standard (CEN/TC 278/WG8) developed for information exchange between traffic management centres developed in line with the European Union ITS Action Plan. Datex II focuses on *events*, any temporary conditions affecting trip planning and traffic management. STREETLIFE adopts this scheme for both describing events and as an external interface for crowdsourced traffic data. It is a multipart standard and is basis for common exchange between the actors of the traffic and travel information sector in European Union. DATEX II defines a common set of data exchange specifications to support the vision of a seamless interoperable exchange of traffic and travel information across boundaries, including national, urban, interurban, road administrations, infrastructure providers and service providers.

Besides crowdsourced data the TRE STREETLIFE pilot receives real-time information about available parking places using DATEX II. Park and Ride routing needs the information, if there are available parking places to switch from private vehicle or bicycle to public transport at park and ride locations. DATEX II includes specification for parking place availability (DATEX II Part 6 Parking Information).

Information exchanged with DATEX II systems is composed of different basic elements:

- Road and traffic related events (called “Traffic elements”)
- Operator actions
- Impacts
- Non-road event information
- Elaborated data (derived/computed data, e.g. travel times, traffic status)
- Measured data (direct measurement data from equipments or outstations, e.g. traffic and weather measurements)

In addition there are also predefined locations and measurement site table information exchanged. These are not part of the basic elements, but are required if the corresponding information in the basic elements is to be understood by a client

5.3.1. Traffic Elements

These are all events which are not initiated by the traffic operator and force him to undertake (re)actions. They are classified in the following main categories:

[2] http://www.datex2.eu/content/standardization

- Abnormal traffic (long queues, stop and go, …)

1. Deviation from schedule (public transport)
2. Cancellation of the run (public transport)
3. Detour (public transport)

- Accidents
- Obstructions:

1. animal presence,
2. vehicle presence,
3. obstructions due to environment (avalanches, flooding, fallen trees, rock falls, …),
4. obstructions due to infrastructure (fallen power cables, …)
5. other obstructions including people Activities (public event, disturbance, …)

- Incidents on equipments or systems (Variable message sign out of order, tunnel ventilation not working, emergency telephone not working, …)
- Conditions : driving conditions related to weather (ice, snow, …) or not (oil, …), conditions related to environment (precipitation, wind, …), …

5.3.2. Operator Actions

Operator actions are classified in four main categories:

- Network management: road closure, alternate traffic, contra flow …

1. traffic control : rerouting, temporary limits

- Roadworks: resurfacing, salting, grass cutting …
- Roadside assistance: vehicle repair, helicopter rescue, food delivery …
- Sign settings: variable message signs, matrix signs.

5.3.3. Impacts

It contains, in particular, information on lane availability and on delays (in seconds, in time range or globally)

5.3.4. Non-road Event Information

It concerns information about events which are not directly on the road: transit service information, road operator service disruption, car parks.

5.3.5. Elaborated Data

These sets of data are normally derived on a periodic basis by the Traffic Centre from input data for specified locations:

- Travel times : elaborated time, free flow time, normally expected time

- Traffic status = attribute with 5 possible values:

1. free flow,
2. heavy,
3. congested,
4. impossible,
5. unknown

- Traffic values (normally published as measured data, but can be derived on a periodic basis and published as elaborated data): flow, speed, headway, concentration and individual vehicle measurements.
- Weather values (normally published as measured data, but can be derived on a periodic basis and published as elaborated data): precipitation, wind, temperature, pollution, road surface condition and visibility.

They can be forecast values.

5.3.6. Measured Data

These data sets are normally derived from direct inputs from outstations or equipments at specific measurement sites (e.g. loop detection sites or weather stations) which are received on a regular (normally frequent) basis. This data matches with our definition of *passive* real-time data sources. However, these events are not intended to be used as hard real-time data, such as immediately transmitted floating GPS measurements, as the data considers rather traffic situation (the world state) than individual dynamic entities.

- Traffic values: flow, speed, headway, concentration and individual vehicle measurements.
- Weather values: precipitation, wind, temperature, pollution, road surface condition and visibility.
- Travel times (normally published as elaborated data, but direct outstation values can be published as measured data): elaborated time, free flow time, normally expected time
- Traffic status (normally published as elaborated data, but direct outstation derived values can be published as measured data) = attribute with 5 possible values (see Elaborated data above).

5.4. Real-time data management

Maintaining a real-time traffic state involves an efficient infrastructure and advanced algorithms to allow scalability in large scale environments, where potentially hundreds of thousands or even million mobile clients send their state updates to services, and also receive updates about the current traffic states. Related research has been carried out in simulated environments in both military applications and games. We draw experience from them and apply suitable optimizations to manage real-world states.

5.4.1. Networked Virtual Environments

In networked virtual environments (NVEs), multiple users and other *dynamic entities* share a common world. NVEs range from text-based games such as *multi-user dungeons* (MUDs) to immersive, *collaborative virtual environments* (CVEs). The most popular form of NVEs is networked 3D gaming, including *massive multi-player online role-playing games* (MMORPG) such as the *World of Warcraft*, and *first person shooters* (FPS) such as the *Counter-Strike*.

The main challenge in NVEs is related to the scalability of the shared environment. There are two common bottlenecks. First, the system may face computational limits. This is typical in client-server systems, where a server is needed for decision making to ensure consistency and persistence of the virtual world. For example, as a near simultaneous action, two clients may claim to occupy the same spot or resource. The physics of the virtual world may also need to be guarded by the server. In this case, the server performs a full world simulation for each client, and eventually saturates as the number of clients increase. Typical cases are first person shooters, which provide high interaction between clients, and where server-to-client traffic increases linearly with the number of clients [32]. The system architecture typically limits the number of simultaneous FPS clients to 32–64.

The second potential bottleneck is the network. On peer-to-peer (P2P) systems, simulation is distributed to the clients, but they need to send their state updates to other clients. As the number of clients increase, so does the network traffic. The very first large scale NVE, the SIMNET, was a combat simulator for small unit interaction. In *SIMNET*, and the follow-up *NPSNET*, *Players* were expected to be *honest*. The Players were responsible for their part of the simulation, submitting their own locally calculated states and effects of interaction, such as hits on enemy vessels, to other Players. To minimize network traffic, SIMNET Players only transmitted state updates when a Player's extrapolated (*dead reckoned*) state (their *Ghost*) exceeded an error threshold in comparison to the real one [33]. In NPSNET-IV, state updates were distributed among all Players, saturating a 10Mbit/s Ethernet with about 300 Players. In addition, each Player had to calculate the Ghosts of all other Players to run the simulation, causing also a computational bottleneck.

To increase scalability of NVE systems, *interest management* was introduced, to filter communication based on interest expressions, for example geographically (area of interest) or functionally (a tank being interested in ground vehicles [34]) . Later NVEs have utilized spatial localization and the concept of communication visibility [35] in message filtering.

5.4.2. Networking and Representation of Real Time Real World Environments

In networked virtual environments, the challenges have been identified to lie in the scalability of a) computational resources and b) networking resources. The focus has been in the state of dynamic entities, such as Players of a military simulation. The presented NVE applications did not include, for example, routing solutions in dynamically varying environmental conditions. In STREETLIFE, multimodal routing is one of the core functionalities, where users and transportation vehicles can be considered dynamic entities, but where the state of the traffic situation and the environment itself (current states of roads, weather, etc.) constitute a *dynamic world state*. Furthermore, in networked games, clients are typically not expected to be honest, and game servers have been required to run full physical simulations for each client

and event. In the real world, such full physical simulation is not applicable, although certain data filtering may take place, i.e. to compensate for GPS inaccuracies or warping in poor satellite visibility conditions. In general, we will expect our crowdsourced measurement data (see Chapter 4) to be *honest.*

In STREETLIFE, we apply the topology of the underlying road network, the *world graph*, to act as spatial skeleton (see Figure 15). Road segments form *edges* and crossings *nodes* on the graph. Edges can hold a lot of information, similar to OSM tags (see Section 4.1.1), depicting pedestrian walkways, highways, rails etc. In the real time case, edges hold current traffic flows, temperatures, iciness etc. We furnish the graph off-line with *static entities* (such as bus stops, landmarks and other points of interest) and on-line with dynamic entities. These entities then reside on a certain edge or node on the graph (in addition to having the usual geo-coordinates). Topology of the graph allows us to perform routing, infer logically proximal events and predict the progress of the traffic situation. Large area states such as weather can be either linked directly to the graph, or to avoid repetition, recovered indirectly. Spatial components of user data such as 'home' and 'work' can be placed on the graph for personalization purposes.

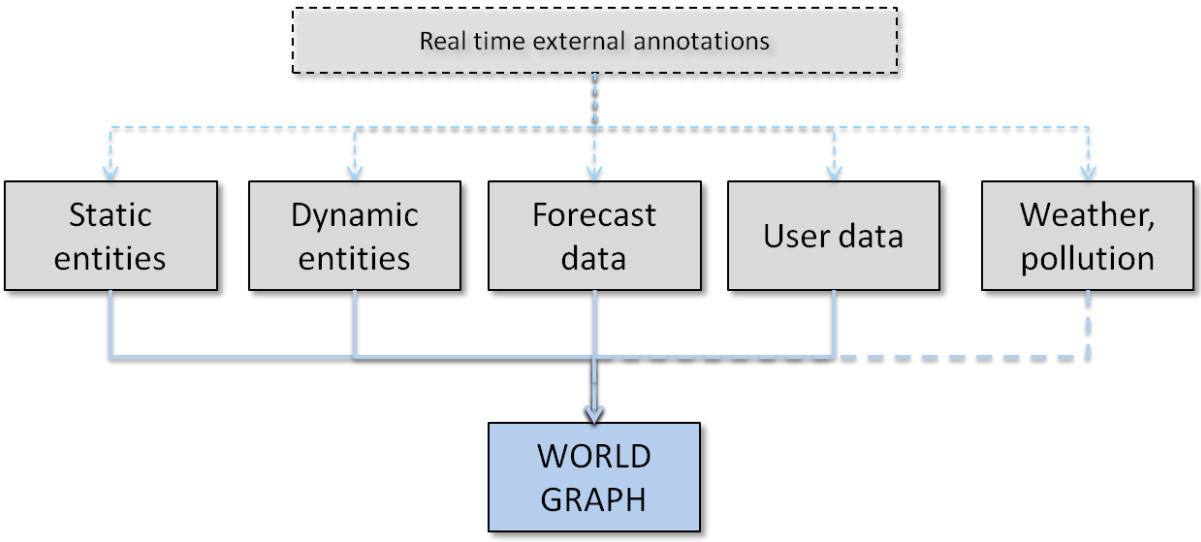

Figure 15. World graph as a topological spatial skeleton for both static and dynamic data.

The World Graph resides at a centralized server, managing the state of a medium-sized city. Scaling up to cover wider regions requires sharing the states between neighbouring World Graph servers. For performance reasons, the real-time state, including the state of the dynamic entities, is held in memory structures for fast access. Various optimizations are applicable to minimize updates to avoid inbound networking congestion. For example, the Player/Ghost paradigm with dead reckoning can minimize dynamic entity updates. Floating GPS transmissions from dynamic entities can be filtered to infer traffic flows, updating the World Graph only when pre-assigned thresholds are exceeded. This functionality can be isolated to *entity proxies*, separate processes that do not stall the actual World Graph management service.

Management of the world state in a centralized server is only the first – and easier – half of the real-time challenge. Given a sufficient amount of mobile clients – dynamic entities – that both transmit their own state and receive the states of other entities, network congestion is unavoidable as the data to transmit increases nearly exponentially along with the number of entities. In the SIMNET system, where only the states of the dynamic entities were propagated from every entity to every entity, a 10Mbit Ethernet was saturated with just 300 Players, despite optimizations. In our case, we would need to propagate the full world state, rendering the situation even worse.

The networking challenge calls for *interest management*, where only the most relevant part of the world state is propagated to the clients. Inferring this relevancy is situational, and can be composed of simple Euclidian distance, topological distance and explicitly or implicitly expressed entities. For example, if our user requests a route, the world state along the route becomes implicitly expressed. If this route involves the use of transport vehicles such as buses, these entities and the state along their current route become similarly expressed.

5.5. Current STREETLIFE Implementations on Real-Time Data

5.5.1. Real-time Data Integration by AALTO

AALTO's work on real-time data management focuses on researching the possibilities regarding multimodal transportation. AALTO uses the OSM data structure and data as the World State skeleton, onto which static data, such as GTFS and real-time data, such as Tampere Public Transportation tracking information. Figure 16 presents the pipeline for external real-time data sets. For scalability purposes, the stages have been separated. Each main block can be considered a proxy, which could reside outside the actual service that maintains the World State.

Real-time inputs are received by special *Input Adapters*, java scripts and web services that parse the input data. This data is then encoded into an internal binary stream to the *Data Correction* stage. Here, the data is mapped to the OSM based road topology. This stage helps also in improving GPS accuracy, forcing our vehicles or pedestrians to roads and walkways. The result is encoded and transmitted to the actual World State service via a binary Server Adapter.

Currently, an Input Adapter for Tampere's SIRI feed has been implemented. Data correction is performed for Tampere's buses, which are mapped onto OSM data. Unfortunately, the data feed is not completely machine compatible, featuring text strings intended for passengers that convey information about where a particular vehicle is going. However, these strings do not match with bus end stops but rather describe areas through which these vehicles go. We are yet to develop an algorithm that uses this information to infer a direction and an exact route of a vehicle. Therefore we are currently only able to track about 50-75% of vehicles. A Server Adapter has been implemented, feeding the updates to our World State service.

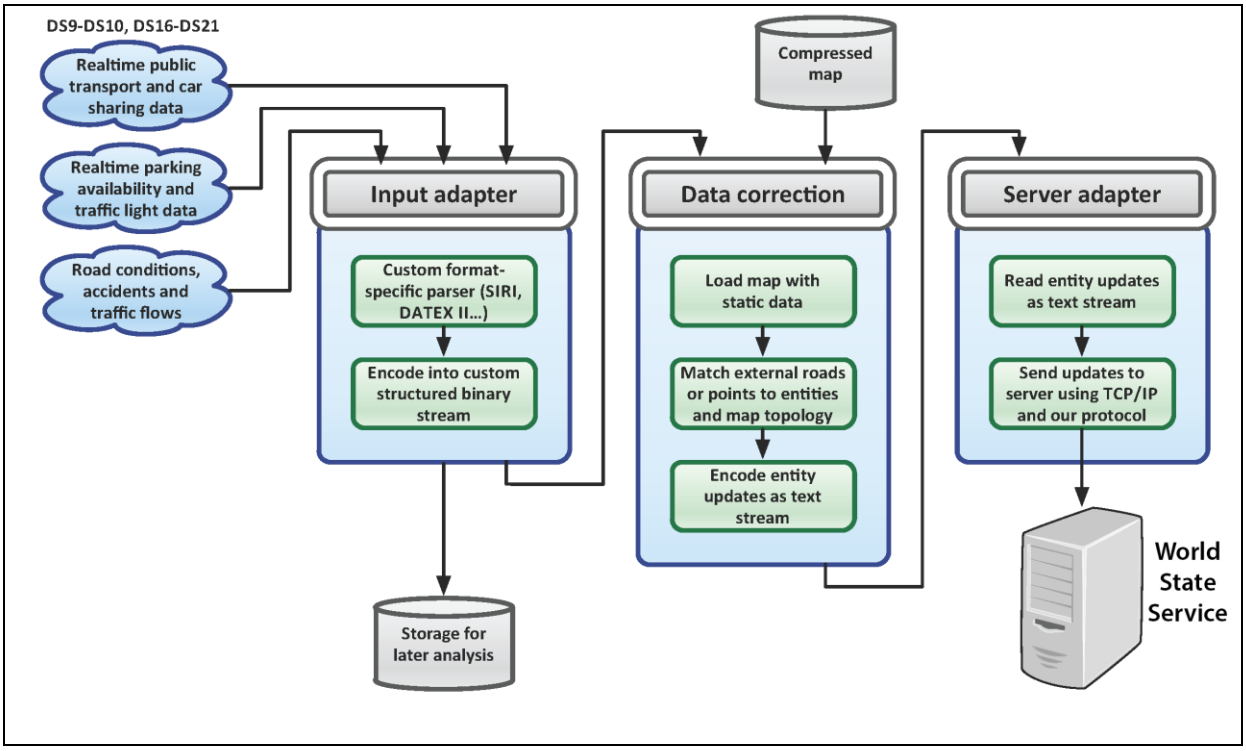

Figure 16. AALTO's Real-time data integration pipeline.

An initial World State Service has been implemented, holding the world state and transmitting updates to clients. An appropriate Interest Management algorithm is being designed. Currently, a very simple data flow culling mechanism is in place, limiting the propagation of vehicle updates to a rectangular area around Tampere city center. A Datex II compatible output is also being implemented for CGI and the TRE Pilot.

5.5.2. Real-time Data Integration by CGI

Key part of the integration of real-time data is efficient processing, data correction mechanisms and ability to match real-time data to other data elements. One way to have a very efficient data management is keeping everything in memory and creating mechanisms to update these memory structures.

For example, the STREETLIFE TRE Pilot uses in-memory structures for the Tampere region passenger information and journey planner (RTPIS). The multi-modal router keeps all the needed static data in memory for fast access. The real-time feeds send updated information to these memory structures, which the multimodal router can take into account if bus has been delayed and user will get up-to-date information for their trip itinerary. In park and ride mode it is important to guide people to the locations where there is parking available. Therefore the real-time router will check the parking availability during the route itinerary calculation.

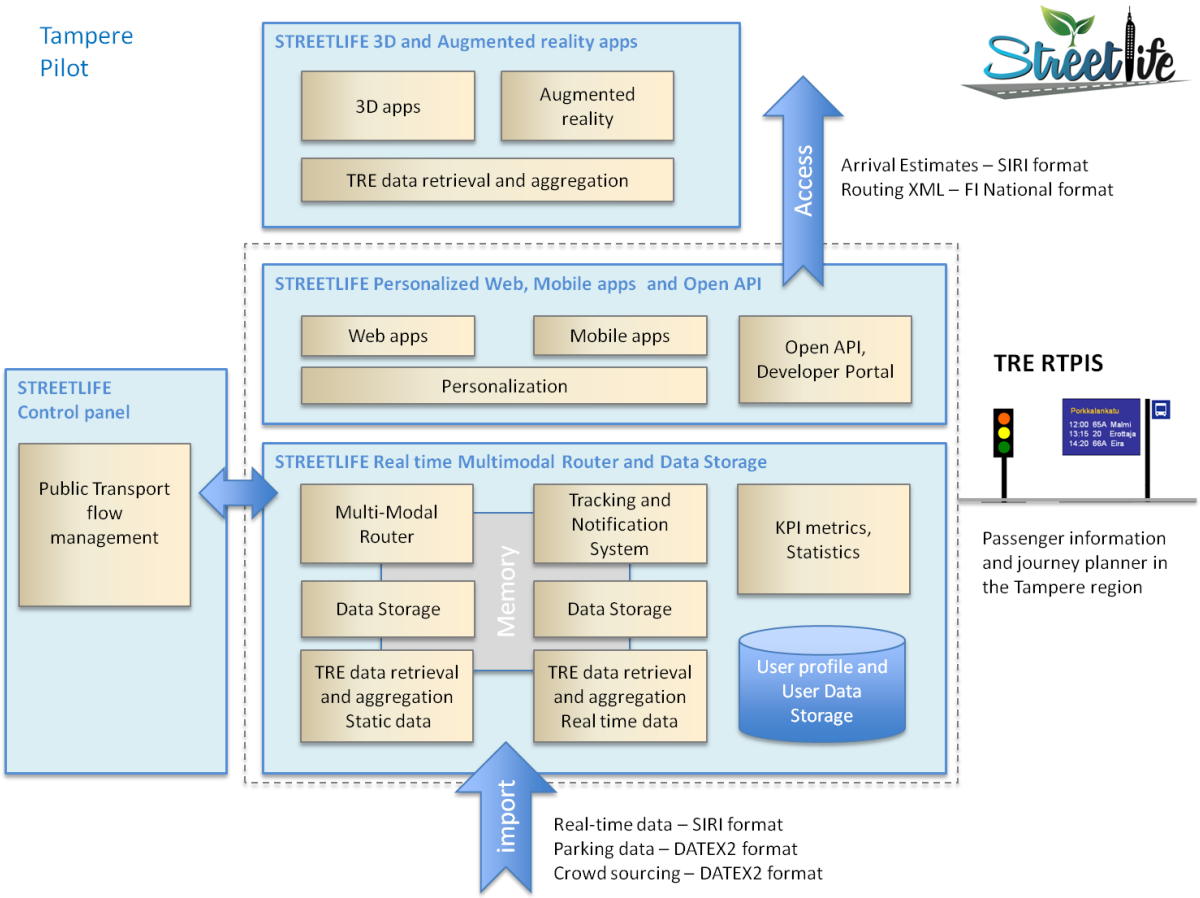

Figure 17: TRE Pilot's Real-time data integration pipeline. Real-time data flows in SIRI and DATEX II formats

The processing of real-time data in STREELIFE TRE pilot is pre-processing as described in the blueprint architecture in D2.2.1, where the router can be affected almost on the fly. The multi-modal router will be able to route all transit model, taking into account real-time events regarding bus arrival estimates and delays or cancellations. In addition real-time router will check the availability of parking places in park and ride locations (in park and ride mode).

6. Open Data Sources and Formats

In Section 2.2.2 we classified the data from the SDM into *STREETLIFE data* and *External data*. The external data, provided by data sources outside STREETLIFE, can be legacy or open data. Legacy data comes from the pre-existing software at the pilot sites. In this chapter we concentrate on the open data and open data sources available for STREETLIFE.

Governments and municipalities are increasingly making public transportation data available on the Internet. As Open Data interfaces, this information is freely accessible without restrictions from copyright, patents or other mechanisms of control [36]. As typical use cases, this data can be used to explore and characterize current and historical service levels or to forecast operations in the immediate future [37]. Opening previously closed data sources or creating such services for the public comes with a cost. However, there are several occasions where there are clear benefits, such as creating more compelling advanced searches, mashups with other data sources and data visualization [37], [38].

6.1. General Transit Feed Specification

The General Transit Feed Specification (GTFS), developed by Google, defines a common format for public transportation schedules and associated geographic information. GTFS "feeds" allow public transit agencies to publish their transit data and developers to write applications that consume that data in an interoperable way. A GTFS feed is composed of a series of text files collected in a ZIP file. Each file models a particular aspect of transit information: stops, routes, trips, and other schedule data.

For STREETLIFE, City of Tampere (through their ITS Factory data service (http://data.itsfactory.fi) and Trentino Province for Rovereto (http://dati.trentino.it/dataset/trasporti-pubblici-del-trentino-formato-gtfs) provide their City bus data in GTFS format, as Open Data. Table 9 presents the available files in GTFS specification.

Table 9. Tampere Public Transport GTFS feed

Description file	Definition
Agency	• Transit agency that provides the data in this feed: Tampereen joukkoliikenne.
Stops	• Individual locations where vehicles pick up or drop off passengers.
Routes	• Transit routes. A route is a group of trips that are displayed to riders as a single service.
Trips	• Trips for each route. A trip is a sequence of two or more stops that occurs at specific time.
Stop_times	• Times that a vehicle arrives at and departs from

	individual stops for each trip
Calendar	• Dates for service IDs using a weekly schedule. Specify when service starts and ends, as well as days of the week where service is available.
Calendar_dates	• Exceptions for the service IDs defined in the calendar.txt file. If calendar_dates.txt includes ALL dates of service, this file may be specified instead of calendar.txt.
Shapes	• Rules for drawing lines on a map to represent a transit organization's routes.

6.2. OSM and GTFS Integration

AALTO has created a process to incorporate OSM and GTFS data to act a) as the primary data set used for routing and b) as a World State skeleton for static and real-time data management. Figure 18 presents the pipeline. An OSM data set covering Tampere region is downloaded and relevant roads along with their tags extracted. Public transit stops and routes are fetched in GTFS format, and extracted. External points are matched to nearest street to place them into the topology. This set is then stored aside in OSM format for review with any OSM tools. Finally, the map is compressed and encoded for efficient storage, transmission and utilization in routing. *OSM Squeezer* is Open Source.

In the next phase of the STREETLIFE project, the pipeline will encompass more data, such as parking locations, traffic light locations and any external data sources that are deemed useful.

The use of OSM in real-time data and in routing is described in Section 4.1.

6.3. Traffic Notification Feeds

Another Open Data source is the traffic situation announcements, especially regarding the public transport operations. In STREETLIFE the most useful Notification messages are the ones which can match to single departure or single journey and therefore enables personalized information to a traveller.

The SIRI interface of the TRE Pilot (see Section 5.2) is a good example of Open Data traffic notifications. The SIRI includes general messages with identification information to affected bus routes or stops. The SIRI general messages can also have messages without identification like “bad winter weather affects public transport, please take weather into account when planning your travel” or even “Merry Christmas”. This service is provided by City of Tampere, free of charge, to any registered users. The SIRI data originates from the Tampere region Real Time Passenger Information System (RTPIS) operated by Logica.

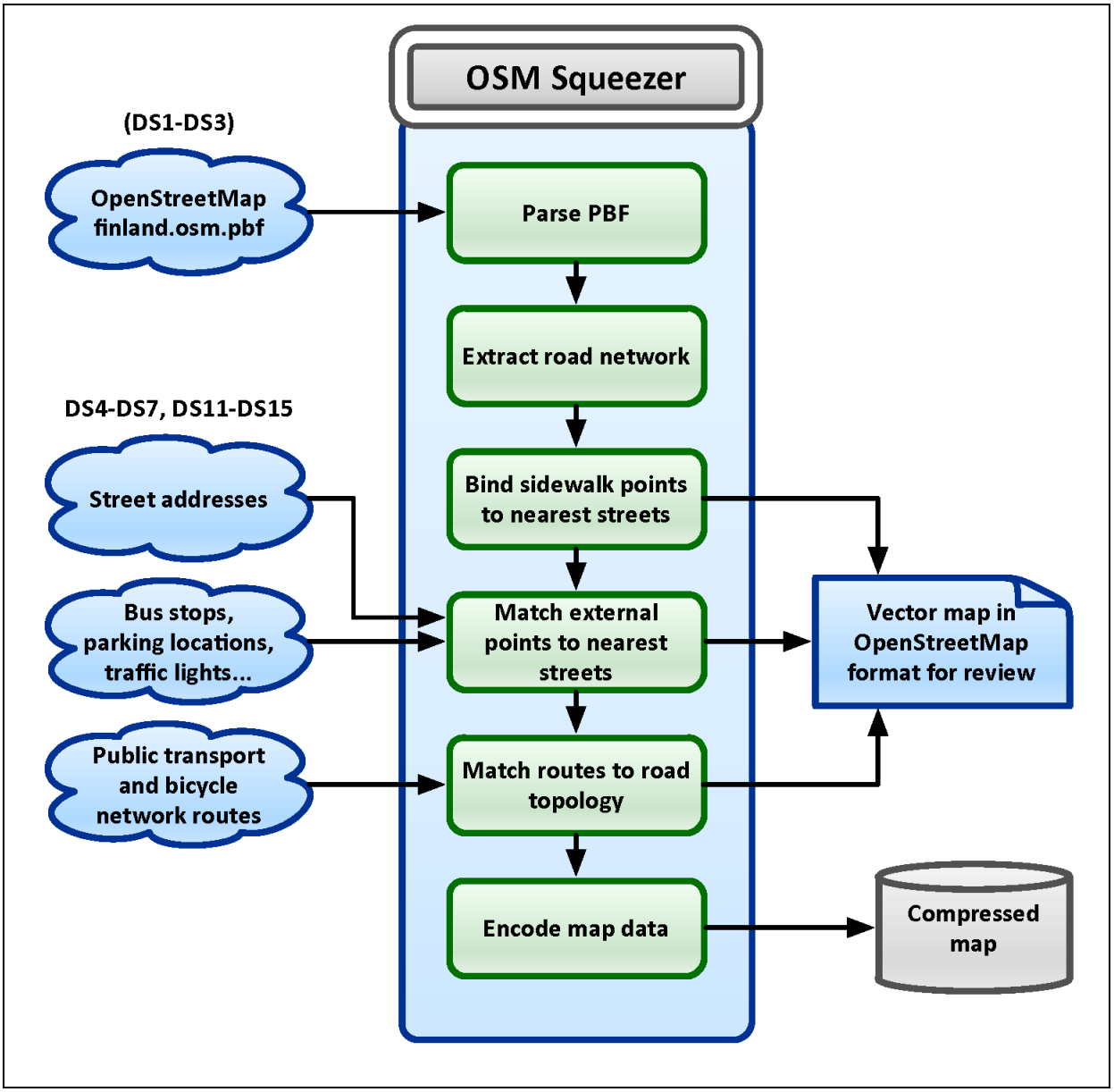

Figure 18. AALTO's *OSM Squeezer*, a pipeline to integrate OSM and GTFS data to STREETLIFE.

7. DATA CORRELATION

Featuring personalisation is one of the key values for the STREETLIFE end-user application developed by WP5. Definitions and personalisation plans in STREETLIFE are given in the Deliverable D5.2.1[24].

From input data perspective, in STREETLIFE we recognize two personalisation possibilities: static data based personalisation and personalisation based on correlation of static with real-time data. Static data based personalisation, for example using user preferences, is going to be included in the basic functionality of the end-user application, initially in the trip planner.

The second type of personalization is used for real-time support of users and it is going to be part of the post-personalisation process defined in D5.2.1 [24]. In this case, beside the static preferences of the user, we plan to include the accumulated knowledge from user's behaviour and the real-time data that is relevant for the current user and situation. This functionality is planned to be part of the data correlation and analysis component of the STREETLIFE data management system, which will be developed during the second project iteration.

Figure 19 presents examples of data correlation used for deriving more complex, aggregated user data that is further used for personalization. In the first example (Figure 19, top) the current transportation mode of the user is derived from several data sources (in this case from the interaction context, traffic state and current position). A research needs to be done to determine the optimal combination of sources that fuse into one attribute.

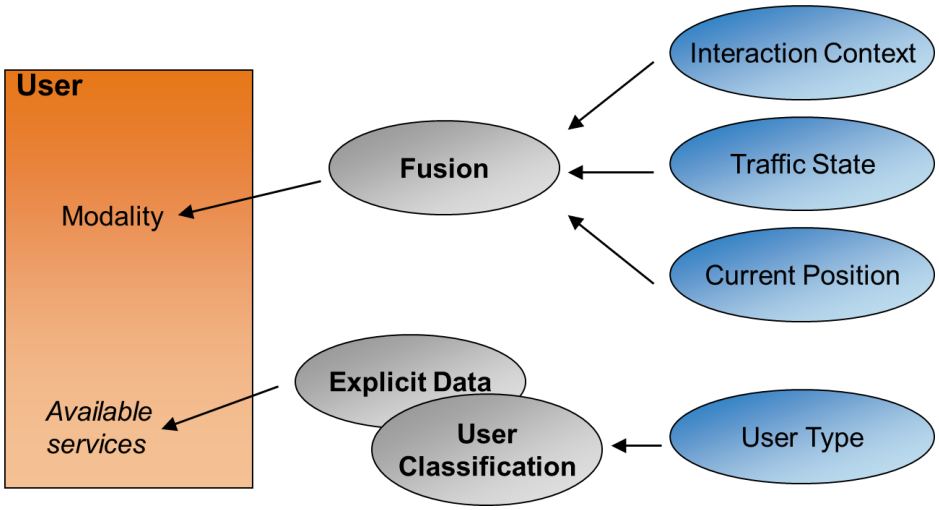

Figure 19: STREETLIFE data correlation example

The bottom part of Figure 19 shows an example of data correlation on static data. In the example, the available services for the user are determined from the user type and her preferences.

WP3 and WP5 will work together on solutions for both cases of personalization. In either of the cases, situation awareness gained by extended knowledge about the user, the vehicle and the current traffic situation is required.

The reasoning methods and techniques will be planned and developed in coordination with the planed scenarios at the pilot sites. The development of these techniques is planned for the second and third year of the project, once the basic functionality of the system is available.

The platform on which these techniques will be developed depends on the pilot site. The research-oriented prototype that is going to be developed in Rovereto (as described in Section 3.2.1) will use KAPcom - Knowledge Management, Adaptation, and Personalization Component [8] as a data correlation and analysis component in combination with Fi-ware. KAPcom is knowledge management platform, initially developed for the in-vehicle domain.

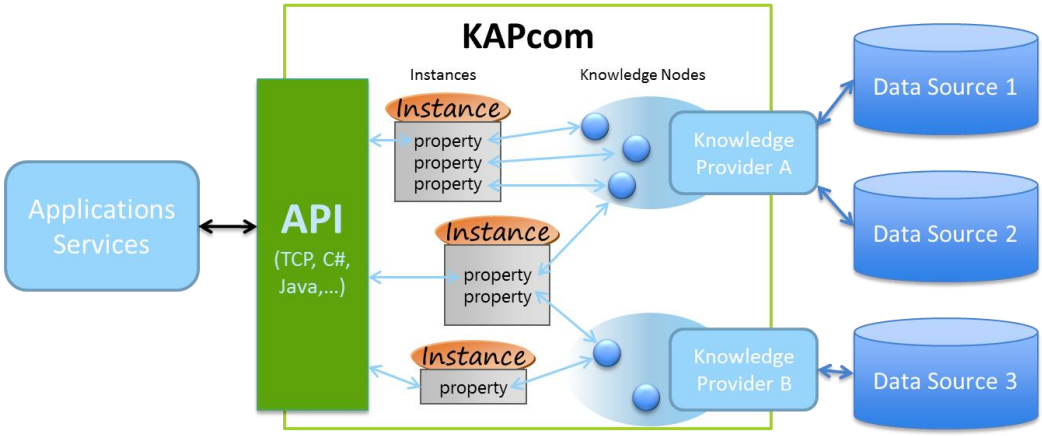

Figure 20: KAPcom components

Figure 20 shows the KAPcom architecture. KAPcom is designed as a middleware between the heterogeneous data sources and the application and services. To the clients, functions, and services, the knowledge base is designed to appear as a seamless space populated with the structured and semantically augmented knowledge that is available. In the background the knowledge is either directly provided or produced by reasoning methods from the available data sources. The knowledge can be either stored or obtained on the fly when queried. This way, external user profiles, inference services, sensors, and even web services can be connected by providing a storage provider implementation for the knowledge manager [8].

Within WP3 we plan to extend the existing solution so it supports the data represented in the SDM. We will also further develop new STREETLIFE specific knowledge providers.

8. CONCLUSION

This deliverable elaborated on different aspects of mobility data integration and the work done within the package WP3 to address the challenges that come with each of the aspects. Represented by separate chapters in the deliverable, the tasks of WP3 work on data definition, knowledge representation, techniques for real-time data management, crowdsourcing techniques and data correlation and analysis.

The second iteration of the STREETLIFE data model was discussed in Chapter 2. The refinement of the initial data model was done in collaboration with the other work packages in the project reflecting the requirements of the STREETLIFE system. The chapter also presented two types of STREETLIFE data categorization. By update interval and update relevance we define three data categories: *real-time data*, *dynamic data* and *static data*. By data origin we define two categories: *STREETLIFE data* and *External data*. Open data, as a very important source for external data is presented in Chapter 7.

The architectural placement of the STREETLIFE data management subsystem is presented in Chapter 3. In addition to the general view of the data management, this chapter elaborate on the three implementations of the architecture at the pilot sites. Separate for Rovereto, Tampere and Berlin we give an overview of the current state of the data management components and discuss the future development within STREETLIFE.

In STREETLIFE we will research the potential of crowdsourcing as a data source in multi-modal route planner. In Chapter 4 we define three main crowdsourced data categories: *Active and Static*, *Active and Dynamic* and *Passive and Dynamic* crowdsourcing. We give general description and STREETLIFE research and implementation state for each of the categories. We present OSM data as an example for Active and Static crowdsourcing, user active participation as an example for Active and Dynamic crowdsourcing and bicycle floating data as Passive and Dynamic crowdsourced data.

In Chapter 5 we analyse different aspects of real-time data management. We elaborate on the different real-time data sources. Real-time tracking and real-time events are elaborated in more detail. We provide an overview on the related work on advanced real-time management algorithms and present the on-going work within STREETLIFE.

The personalisation aspect of the STREETLIFE system, seen from the data perspective is discussed in Chapter 7. In this deliverable we provide a high-level discussion on data correlation and analysis for personalisation purposes. We present the data management software KAPcom that is going to be used and further developed as a data correlation component in the project.

APPENDIX A: LITERATURE

[1] M. Mitrevska and S. Castronovo, "D3.1 – Conceptual models and specifications for mobility data integration." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 02-Jul-2014.

[2] B. Dittwald, J. Kohlmorgen, K. van Bernum, B. Schünemann, B. Obst, V. Brandstetter, R. Kelpin, D. Frigeri, N. Perri, F. Avesani, A. Merigo, and C. Schwarz, "D4.2.1 – Mechanisms and tools for mobility management and emission control (initial)." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 10-Jan-2014.

[3] P. Gilka, D. Frigeri, A. Merigo, F. Avesani, and M. Vuorio, "D8.1.1 – Evaluation Plan (initial)." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 10-Jan-2014.

[4] S. Cuno, L. Farid, F. Thiemer, G. Valetto, M. Bahr, V. Brandstetter, B. Obst, T. Schilling, S. Schaffer, A. Nurminen, and M. Vuorio, "D2.2.1 – STREETLIFE Blueprint Architecture, Security Architecture, and site-specific architectures (initial)." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 10-Jan-2014.

[5] L. Farid, F. Thiemer, S. Cuno, B. Dittwald, D. Gorny, G. Valetto, A. Marconi, M. Bahr, B. Obst, V. Brandstetter, D. Frigeri, F. Avesani, S. Schaffer, and M. Vuorio, "D2.1 – Report on gap analysis and incentive models." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 06-Feb-2014.

[6] "FIWARE » Open APIS for Open Minds." [Online]. Available: http://www.fi-ware.org/. [Accessed: 23-Sep-2014].

[7] "FI-PPP | Future Internet PPP." [Online]. Available: http://fi-ppp.eu/. [Accessed: 23-Sep-2014].

[8] M. Feld, "A speaker classification framework for non-intrusive user modeling: speech-based personalization of in-car services," 2012.

[9] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Comput. IEEE*, vol. 7, no. 4, pp. 12–18, 2008.

[10] "Publish/Subscribe Context Broker - Orion Context Broker | FIWARE Catalogue." [Online]. Available: http://catalogue.fi-ware.org/enablers/publishsubscribe-context-broker-orion-context-broker. [Accessed: 17-Sep-2014].

[11] O. M. Alliance, "OMA Web Services Enabler (OWSER): Overview," *Approv. Version*, 2004.

[12] S. Hoffman, *Apache Flume: Distributed Log Collection for Hadoop*. Packt Publishing Ltd, 2013.

[13] "BigData Analysis - Cosmos | FIWARE Catalogue." [Online]. Available: http://catalogue.fi-ware.org/enablers/bigdata-analysis-cosmos. [Accessed: 23-Sep-2014].

[14] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[15] A. Hadoop, *Apache Hadoop*. 2011.

[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 2010, pp. 1–10.

[17] "Complex Event Processing (CEP) - IBM Proactive Technology Online | FIWARE Catalogue." [Online]. Available: http://catalogue.fi-ware.org/enablers/complex-event-processing-cep-ibm-proactive-technology-online. [Accessed: 23-Sep-2014].

[18] D. Frigeri, R. Kelpin, T. Pezzato, F. Thiemer, G. Valetto, M. Vuorio, and M. Wollina, "D6.1 Specification of city pilots for the first STREETLIFE operation and evaluation." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 02-Apr-2014.

[19] T. Roopa, A. N. Iyer, and S. Rangaswamy, "CroTIS-Crowdsourcing Based Traffic Information System," in *Big Data (BigData Congress), 2013 IEEE International Congress on*, 2013, pp. 271–277.

[20] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *Commun. Mag. IEEE*, vol. 49, no. 11, pp. 32–39, 2011.

[21] K. D. Giudice, "Crowdsourcing credibility: The impact of audience feedback on Web page credibility," *Proc. Am. Soc. Inf. Sci. Technol.*, vol. 47, no. 1, pp. 1–9, 2010.

[22] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels, "Citizen Noise Pollution Monitoring," in *Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections Between Citizens, Data and Government*, Puebla, Mexico, 2009, pp. 96–103.

[23] H. Weinschrott, F. Dürr, and K. Rothermel, "StreamShaper: Coordination algorithms for participatory mobile urban sensing," in *2010 IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2010, pp. 195–204.

[24] A. Nurminen, S. Schaffer, and A. Marconi, "D5.2.1 – End-user applications techniques and tools." EU project "STREETLIFE – Steering towards Green and Perceptive Mobility of the Future," 10-Jan-2014.

[25] "Waze: Free Community-based Mapping, Traffic & Navigation App." [Online]. Available: https://www.waze.com/. [Accessed: 30-Sep-2014].

[26] R. Sen, "RasteyRishtey: A social incentive system to crowdsource road traffic information in developing regions," in *Mobile Computing and Ubiquitous Networking (ICMU), 2014 Seventh International Conference on*, 2014, pp. 171–176.

[27] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee, "TruCentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 160–166.

[28] J. L. Berggren, A. Jones, and others, *Ptolemy's Geography: an annotated translation of the theoretical chapters*. Princeton University Press, 2000.

[29] "Protocol Buffers — Google Developers." [Online]. Available: https://developers.google.com/protocol-buffers/. [Accessed: 27-Sep-2014].

[30] L. P. Deutsch, "DEFLATE compressed data format specification version 1.3," 1996.

[31] SIRI, "SIRI - Service Interface for Real Time Information." CEN/TS 15531(prCEN/TS-OO278181), 2011.

[32] J. Färber, "Tra_c modelling for fast action network games." Multimedia Tools Appl, 23(1):31–46, 2004.

[33] B. Blau, C. E. Hughes, M. J. Moshell, and C. Lisle, "Networked virtual environments," in *Proceedings of the 1992 symposium on Interactive 3D graphics*, 1992, pp. 157–160.

[34] M. R. Macedonia, "A network software architecture for large scale virtual environments," Monterey, California. Naval Postgraduate School, 1995.

[35] M. Capps and S. Teller, "Communication visibility in shared virtual worlds," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997. Proceedings., Sixth IEEE Workshops on*, 1997, pp. 187–192.

[36] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[37] K. Kuhn, "Open Government Data and Public Transportation," *J. Public Transp.*, vol. 14, no. 1, 2011.
[38] D. Robinson, H. Yu, W. P. Zeller, and E. W. Felten, "Government data and the invisible hand," *Yale JL Tech*, vol. 11, p. 159, 2008.