<table>
<thead>
<tr>
<th>Project number:</th>
<th>288577</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym:</td>
<td>UrbanAPI</td>
</tr>
<tr>
<td>Project title:</td>
<td>Interactive Analysis, Simulation and Visualisation Tools for Urban Agile Policy</td>
</tr>
<tr>
<td>Instrument:</td>
<td>STREP</td>
</tr>
<tr>
<td>Call identifier:</td>
<td>FP7-ICT-2011-7</td>
</tr>
<tr>
<td>Activity code:</td>
<td>ICT-2011.5.6 ICT Solutions for governance and policy modelling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start date of Project:</th>
<th>2011-09-01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration:</td>
<td>39 month</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliverable reference number and title (as in Annex 1):</th>
<th>D3.5 Data Integration Components Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date of deliverable (as in Annex 1):</td>
<td>30.11.2012</td>
</tr>
<tr>
<td>Actual submission date:</td>
<td>see “History” Table below</td>
</tr>
<tr>
<td>Revision:</td>
<td>2</td>
</tr>
</tbody>
</table>

**Organisation name of lead contractor for this deliverable:**

Fraunhofer IGD

---

**Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)**

<table>
<thead>
<tr>
<th>Dissemination Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>PP</td>
</tr>
<tr>
<td>RE</td>
</tr>
<tr>
<td>CO</td>
</tr>
</tbody>
</table>
Title:
Data Integration Components Documentation

Author(s)/Organisation(s):
Jens Dambruch / Fraunhofer IGD, Jan Peters-Anders / AIT, Ernst Gebetsroither / AIT

Working Group:
WP3 – Fraunhofer IGD, AIT

References:
Description of Work, D2.1, D3.2, D3.3, D4.1

Short Description:
A description on the components used to integrate the data for the urbanAPI applications. It explains the approach to integrate the data provided by the cities into the UrbanAPI applications. The basic approach to integration with CityServer3D is outlined as well as integration of GSM and urban-growth data.

Keywords:
Interoperable, integration, data, GIS

History:
<table>
<thead>
<tr>
<th>Version</th>
<th>Author(s)</th>
<th>Status</th>
<th>Comment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Jens Dambruch</td>
<td>New</td>
<td>Initial draft and table of contents</td>
<td>5.11.2012</td>
</tr>
<tr>
<td>004</td>
<td>Jens Dambruch</td>
<td>Rfc</td>
<td>Rework after comments from Joachim Rix</td>
<td>11.01.2013</td>
</tr>
<tr>
<td>005</td>
<td>Ernst Gebetsroither</td>
<td>Rfc</td>
<td>Rework after comments from Joachim Rix</td>
<td>11.01.2013</td>
</tr>
<tr>
<td>006</td>
<td>Jan Peters-Anders</td>
<td>Final</td>
<td>Rework after comments from Joachim Rix</td>
<td>14.01.2013</td>
</tr>
<tr>
<td>007</td>
<td>Jens Dambruch</td>
<td>Rfc</td>
<td>Rework after comments from Wolfgang</td>
<td>25.01.2013</td>
</tr>
<tr>
<td>Version</td>
<td>Reviewer</td>
<td>Comment</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>Jan Peters-Anders</td>
<td>Final Rework after comments from Wolfgang Stemberger and Wolfgang Loibl</td>
<td>31.01.2013</td>
<td></td>
</tr>
<tr>
<td>009</td>
<td>Jens Dambruch</td>
<td>Final Rework after comments from Wolfgang Loibl and Wolfgang Stemberger</td>
<td>08.02.2013</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>Ernst Gebetsroither</td>
<td>RFC Updated Chapter 4</td>
<td>26.08.2014</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>Ernst Gebetsroither</td>
<td>Final Rework after comments from Wolfgang Loibl and Maria Lemper</td>
<td>11.09.2014</td>
<td></td>
</tr>
</tbody>
</table>

**Review:**

<table>
<thead>
<tr>
<th>Version</th>
<th>Reviewer</th>
<th>Comment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>006</td>
<td>Wolfgang Stemberger, Wolfgang Loibl</td>
<td>See Review Form of Deliverables, UrbanAPI QA-D3.5_Data Integration Components Documentation_FINAL.doc submitted on 24.1.2013</td>
<td>24.01.2013</td>
</tr>
<tr>
<td>008</td>
<td>Wolfgang Loibl</td>
<td>Some minor changes, see e-mail of 08.02.2013</td>
<td>7.2.2013</td>
</tr>
<tr>
<td>009</td>
<td>Wolfgang Loibl, Maria Lemper</td>
<td>See Review Form of Deliverables, UrbanAPI QA-D3.5_Data Integration Components Documentation_second version_FINAL</td>
<td>7.2.2013</td>
</tr>
</tbody>
</table>
About this Document ........................................................................................................................................ 5
1 Introduction .................................................................................................................................................. 5
2 The 3DVR Application and CityServer3D .................................................................................................. 5
  2.1 Requirements for a Geo Information System ......................................................................................... 6
  2.2 The Architecture of an interoperable Geo Information System ............................................................... 7
    2.2.1 Interfaces Layer .............................................................................................................................. 8
    2.2.2 Functional Layer ............................................................................................................................ 8
    2.2.3 Data Access Layer ......................................................................................................................... 8
    2.2.4 Metamodel Layer ........................................................................................................................... 9
    2.2.5 Implementation as CityServer3D ..................................................................................................... 10
  2.3 Special Developments – The CityGrid-Data-Access- Component for CityServer3D ............................... 11
3 Motion Explorer Data .................................................................................................................................. 13
  3.1 Import of Pre-processed Data .............................................................................................................. 17
  3.2 First resulting Data Structures ............................................................................................................ 18
  3.3 Data Export Functionalities ............................................................................................................... 18
4 Urban Development Simulator Data ......................................................................................................... 20
  4.1 Pre-processing of tabular Data ........................................................................................................... 20
  4.2 Handling of Geo-Data ......................................................................................................................... 22
About this Document

This document deals with the software components to integrate the data provided by the cities into the UrbanAPI applications. According to the description of work for UrbanAPI [1] it is the deliverable D3.5 – Data Integration Components of work package WP3: Software Platform Development: Database, Integration, Visualisation and Simulation Tools. This is the final version of the document after development has finished. Only chapter 4 was updated for the final version.

1 Introduction

Urban planning involves a lot of geospatial data, from cadastral data to infrastructure plans like the ones for streets or electricity lines, 3D-models of buildings or data from aerial laser scans. This data shall be used to enable a ICT governance of the city. The goal is to have an integrated view on aspects covered by that data. This means not only to consider one aspect like infrastructure planning but also to assess impacts on other topics and fields for planned actions. In the following chapters our approach and the tools that will be used for this integration will be explained.

The next step after processing and harmonizing the data in work package 4 is to integrate it in the UrbanAPI applications. For the 3DVR application the CityServer3D of Fraunhofer IGD contributes the platform for data integration. Special developments, customization and configuration of CityServer3D are also part of the project and driven by Fraunhofer IGD. This is documented in chapter 2.

The components of AIT to integrate the mobile device data is described in chapter 3 and the components of AIT to integrate the Urban-Growth-Modelling data is described in chapter 4.

The UrbanAPI applications will then be used to visualize and analyse the integrated data sets of the partner cities. Software components that are developed for the data harmonisation tasks in WP4 are or will also be documented here.

2 The 3DVR Application and CityServer3D

Spatial data is growing in volume and gains more and more importance over the last years. The heterogeneity among the data formats is an obstacle for the efficient usage of it. For example two municipalities want to cooperate in some city planning task or want to do some analytical explorations (for example traffic between both municipalities) and they are using different software tools with different data formats. The matching of data and a harmonisation on the semantic layer is an involved task if this is done manually. So an interoperable geographic information system is a crucial point for a geo-data-infrastructure. The CityServer3D provided by Fraunhofer IGD is an interoperable Geo Information System (GIS) based on an object oriented architecture derived from experiences with 3D-city-models and their applications. In the UrbanAPI project it is especially suited for the processing and maintenance of 3D City-Data and thus is used as a platform for data management in the 3DVR application.

An example for data processing is the generation of a 3D city-model based on data supplied by a city. The city of Vitoria-Gasteiz supplied 2D footprints of the buildings in the city. Also laser-scan data of the area was supplied, which can be used to determine the height of an object on the ground. For every footprint a shape with the coordinates can be extracted and for each of the coordinates the height above ground could be found by a query on the laser-scan data. Now it is possible to do an extrusion of the footprint to the height from the laser-scan data (see Figure 1). This extrusion results in a box like shape, or to be more precise in a
polyhedron\(^1\). These polyhedra can now be used to create the 3D city-model with the level-of-detail\(^2\) 1. All these operations are directly supported by the CityServer3D.

![Diagram](image)

**Figure 1**: Example for the integration of 2D building footprints with height information from laser scan data. The green arrows indicate the measured height of a point. A 3D model on LOD 1 is the result.

### 2.1 Requirements for a Geo Information System

Thorsten Reitz (2005) [2] analysed architecture approaches for interoperable geographic information systems and gave a survey on the requirements for such a system. A geographic information system (GIS) is defined as a computer based system which manages (processes, stores) spatial data. The data in the GIS can serve multiple different purposes such as planning, (power lines, sewage, traffic, logistics) or marketing for tourism. The systems can be classified by their architecture: Geo-DBMS, Desktop-GIS or server based solutions. In the context of UrbanAPI only the server based approaches will be considered as this naturally fits with the web based applications that are developed in the project.

Thorsten Reitz conducted also essential requirements engineering for a GIS and names the following requirements:

1. A 3D solid shape with flat surfaces and straight edges. (http://en.wikipedia.org/wiki/Polyhedron)
2. In the context of 3D city-models the level-of-detail (LOD) states how detailed a model is. Starting from level 0 to 5 the higher level always includes the lower levels:
   - LOD 0: A terrain model without any buildings
   - LOD 1: Simple box like buildings
   - LOD 2: Buildings with roof shapes and simple textured facades
   - LOD 3: Architectural modelled buildings
   - LOD 4: Interior design
• Interoperability
  Support for standardized interfaces and data formats.

• Flexibility
  The GIS must be easily adaptable and extendable to different usage scenarios.

• Portability
  The GIS must be portable for different platforms.

• Performance
  The GIS shall use state-of-the-art algorithms and technology and avoid overhead.

• Encapsulation
  A component of the GIS should encapsulate complexity and not expose it to the outside to be able to reuse the component easily.

• Low coupling between components
  The architecture of the GIS should be modular and dependencies between components should be minimal.

• Clear architectural separation of concerns
  The paradigms of object oriented programming3, design patterns4 like separation of business logic, user interface logic and data storage must be considered.

Many of these requirements contributed over several years to the development of a prototype that led to the CityServer3D. The software architecture and design that is behind this will be surveyed in the next section.

2.2 The Architecture of an interoperable Geo Information System

Thorsten Reitz [2] describes an architecture of an interoperable GIS considering the requirements named before. A simplified version of the architecture of this modular interoperable GIS is shown in Figure 2. It is divided in several layers. The components and categories are:

• Metamodel (orange box): The Metamodel is the foundation and provides a common programming interface for all components of the system

• Categories of usages (yellow boxes) – The Metamodel serves as a hub for data source adaptations, model for processing logic and serves standardized interfaces

• Characteristic domains, protocol layers or connectors uses (green boxes)

• Practical applications (grey boxes) – Concrete applications of the categories where the Metamodel is used, e.g. implementation of W3DS or importer for ASCII-Grids

---

3 See also Bertrand Meyer [4]
4 See also Gamma et al. [3]
Originally Reitz [2] describes the architecture as modular and distributable, so that modules can run on remote machines. For the prototypes in the UrbanAPI context this is irrelevant. The GIS in this context combines all aspects to a single server solution.

### 2.2.1 Interfaces Layer

The interface layer is responsible for processing incoming requests including authentication of users. In order to be interoperable the support of standardized interfaces is essential. For example the Open Geospatial Consortium (OGC)\(^5\) defines a lot of standards like the WebMapService. Also data formats like CityGML\(^6\) are standardized by the OGC. These are supported by a wide range of commercial applications and research institutions.

### 2.2.2 Functional Layer

The functional or processing layer represents the services and processing logic of the GIS. An example is a function that takes some geometry and creates a new geometry with this, e.g. extrusion of a polygon to a polyhedron. The extrude functionality was used to create simple buildings as boxes based on the footprints of the building. Processing functionality on this layer can be exposed to the interface layer and thus is usable from the outside of the GIS.

### 2.2.3 Data Access Layer

The purpose of the Data Access Layer is to enable data access to diverse data sources from processing actions. Data sources can be for example services like the WebFeatureService, files on the local disk like Shapefiles or data in a relational database. This is an useful abstraction that hides any special properties in the data access. The processing logic needs not to deal with network transfers or local file transfers or databases connections. The processing can abstract from this and concentrate on the actual data to process. For example the transformation of data from a spatial reference systems to another is handled in

---

\(^5\) [http://www.opengeospatial.org/](http://www.opengeospatial.org/)

\(^6\) [http://www.opengeospatial.org/standards/citygml](http://www.opengeospatial.org/standards/citygml)
the data-access layer. Thus an application that accesses data via the data access layer needs not to deal with this.

2.2.4 Metamodel Layer

The name Metamodel can be quite misleading as the Metamodel in this case is not a “model of a model” as the usual naming in computer science would suggest. The term “meta”\(^7\) is more to be understood as a model that is “beside” another model or, as it is used here, “beside all” other models. It is rather to be used as model that can represent the data from other data formats and covers aspects common to all formats supported.

The Metamodel is the core layer of the GIS and covers two essential purposes. First it is the basic tool-kit for all functions in the functional layer. All algorithms in this layer use the common objects of the Metamodel instead of using the data types of a data source.

Second all data objects instantiated as types of the Metamodel are used as intermediate transfer-objects from one data format to another data-format. In case of a data transformation if both the source data format and the target data format have some common elements a simple automated conversion between both formats is possible. Therefore the Metamodel supports all essential data types of the most common formats.

Spatial reference systems are also modelled as central parts of the Metamodel. Conversion between different spatial references is not supported. The data that is imported must be converted in the same reference system before it is integrated. This is done in the data harmonization tasks and documented in deliverable 4.1 about data harmonization.

Figure 3 shows an overview of all classes and packages of the Metamodel. The Metamodel can represent various types of data:

- **Layers**
  A hierarchical grouping of features

- **Features**
  A typical feature can be a building or a set of features that make up a building, like walls and roofs

- **Model**
  Spatial representations of a feature, 2D and 3D geometry including terrains

- **Topological primitives**
  Triangles, faces, edges – a description on how points are connected to each other

- **Meta information and Semantics**
  Every geometry can have additional information. The semantics and meta informations framework behind the Metamodel is extendable and can be adapted.

\(^7\) [http://en.wikipedia.org/wiki/Meta](http://en.wikipedia.org/wiki/Meta)
Finally Thorsten Reitz [2] describes the prototype of an interoperable GIS which has led to the development of CityServer3D together with the team of the Geo Spatial Information Management department at Fraunhofer IGD. The CityServer3D is the result of the considerations and prototypes and adheres to the architecture outlined in the previous sections. The data formats supported by CityServer3D are given in Table 1. Based on the architecture above it is easy to develop a custom component to integrate data in a non-standard format and adapt the GIS to specific project needs. This is the major reason why CityServer3D is the basis for the development of the 3DVR application for UrbanAPI.

![Figure 3: Overview of all classes and packages of the Metamodel. Source: Thorsten Reitz 2005, page 87, [2]](image)
<table>
<thead>
<tr>
<th>Format</th>
<th>Import</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityGML 1.0</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>NAS/ALKIS</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>DXF/DWG</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Shapefiles</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>GeoTIFF</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>DGM(ESRI TIN, ESRI Grid)</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>XYZ/ASC</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>Waypoint+</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>Garfile</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>VRML</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>X3D</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>GeoSciML</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>PCI+RAW</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>KML/KMZ</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Table 2: Supported formats of CityServer3D, all these can be used as input to the UrbanAPI database without additional development efforts.

2.3 Special Developments – The CityGrid-Data-Access- Component for CityServer3D

The city of Vienna uses the CityGrid solutions of UVM-Systems [5] for the management of their 3D-city-model. This commercial software is a closed solution as it only offers to export the data contained in the system in a proprietary XML-format that cannot be converted to a standard format with standard tools. Since it is XML-based it is possible to analyse the proprietary format and map it to the Metamodell. With this mapping it is possible to create some component in the data access layer of CityServer3D, which seamlessly integrates the data of the city of Vienna in the CityServer3D software platform. The component is only used to import the Vienna data as the other city-partners in the project use common standardized formats.
The CityGrid file format is based on well-known concepts from the computer graphics domain. It uses boundary representations to model solid objects. Surfaces consist of points in the 3D space—i.e. vertices. The connection between these points is made by an index. This concept can also be found in other file formats such as VRML or X3D. Apart from that, the CityGrid file format contains metadata such as information about an object’s type or its usage.

The CityServer3D is already able to parse file formats from the computer graphics domain (VRML, X3D, etc.). Parsing CityGrid files can therefore be implemented by reusing existing components. The CityServer3D has a modular architecture which allows for plugging in new data importers (Figure 2).
3 Motion Explorer Data

The Mobility Explorer application carried out by AIT will be based on mobile device location data and socio-economic data imported from proprietary file formats (e.g. ESRI shapefiles, A1 binary log files, text files, MS Excel files etc.). The A1 mobile device location data requires to be heavily pre-processed in order to be applied in the Motion Explorer database.

The most important requirement is to ensure privacy of the contained (already anonymised) mobile device location data, which allows to track individual trips within a certain location accuracy) and to remove location errors. As a consequence, the Motion Explorer application will have only limited possibilities to upload data to the database. Individual mobile device location and movement data are not allowed be uploaded to the system, as only aggregated data – summarised for a raster of square cells to avoid interfering privacy - can be published.

The section below describes the mobile device location data that have been delivered to the urbanAPI project (showing also the huge diversity of the mobile device location data available):

- A1 telecom data: during the first months of the project the data sets delivered by the A1 telecom company differed dramatically in format, content and quality which had to be dealt with by identifying the logic and errors within the datasets. Data delivered later showed better quality in data structure and compression logic. An example of the necessary data analyses and transformations to integrate the data into urbanAPI’s Mobility Explorer is depicted in Figure 4 and Figure 5. Figure 4 shows the pre-processing results (Mobile device users aggregated at 1km raster cells). For a detailed description of the aggregation process see D4.1 - Integrated and Harmonised Data.

![Figure 4: Data integration example: Visualisation of primary data of A1 GSM movement data derived from a generic binary log file converted to a GIS dataset showing a subset of 3000 anonymous IDs movements.](image)

13/23
Figure 5: Data integration example: Visualisation of derived data of A1 GSM movement data: Aggregation of users based on a 1km raster grid and a time stamp.
Telefonica data: The initial datasets from the Spanish Telefonica company for the Vitoria-Gasteiz region are standard CSV tables which show geographic coordinates of the beginning and end of phone calls or text transfer. Here it is also necessary to interpolate the location for certain useful time steps and then to aggregate the data to raster cells (Figure 6).

Figure 6: Data integration example: Visualisation of primary data (tables as text files converted to a point layer) received from Telefonica Spain for Vitoria Gasteiz showing GSM antenna positions and a user’s coordinates when initiating (init) and ending (end) a call on Monday May 16th, 2011
TIM data: The datasets delivered by TIM (Italia) for Bologna Region arrived in early January 2013 and were projected from the proprietary national coordinate system to geographical coordinates (WGS84) which makes necessary are to LAEA/ETRS89 (the projection system used in UrbanAPI). The data description states:

“The traffic files contain the information on the number of active users in each TIM network cell. This number is associated with the pixels of the city of Bologna considering both the radioelectric coverage of each cell, calculated through the TIM network planning software tool, and the probability of having some user traffic on each pixel.”

As far as the TIM’s test data sets could be read until now, the files contain the number of telephone users within certain raster cells. The time stamp is delivered too, which allows mapping of mobile device user densities for time slices after some spatio-temporal interpolation, which allows to shift locations according to defined time steps (Figure 7).

![Figure 7: Data integration example: Depiction of primary data (proprietary ASCII raster) received from TIM showing a heat map of the number of GSM users in Bologna on May 4th, 2012, 16:30 to 17:00](image)

The pre-processing of this data is described in deliverable D4.1 “Integrated and Harmonized Data” chapters 5.2, 6.2 and 7.2. As parts of these newly arrived data without proper coordinates show varying record structure and formats over time, which allows until now no definition of final rules on integrating these data.

---

8 Source: FileDescription_UrbanAPI_20121119_eng.doc, accompanying file to the data sets delivered by TIM in January 2013
3.1 Import of Pre-processed Data

The Motion Explorer application uses PostgreSQL as its backing database. In order to load the pre-processed data an UrbanAPI administrator has to be contacted to do this on the UrbanAPI database server. The administrator will use a command line tool like PostgreSQL’s psql\(^9\) terminal or graphical tools like pgAdmin III\(^{10}\) to load the datasets as tables into the Mobility Explorer database. To ensure good performance during application requests to the database PostgreSQL functions have been implemented within the database to make use of the built-in Query Optimizer (see below for an example of a PostgreSQL function implemented for UrbanAPI showing the creation of so called time slices of the aggregated movement patterns for Vitoria-Gasteiz).

```sql
DECLARE
    tblVar text := 'myTable';
    count int4 := 21759720;
    result int4 := 0;
    minutes int4 = 0;
    ts_minutes int4 = 21758280;
    tblName text = relname || '_slices';
    --relname_debug text = 'night_user_from_lkme4791n2810_during_day_the_geom';
BEGIN
    --EXECUTE 'ALTER TABLE '|| quote_ident(schema) ||'.'|| quote_ident(relname) ||' ADD COLUMN the_geom_4326 geometry;';
    EXECUTE 'CREATE TABLE '|| quote_ident(tblName) || ' (count int4, cellcode varchar, timestamp timestamp)';
    BEGIN
        LOOP
            result := result + 1;
            --tblVar := tblName || to_char(ts_minutes, '99999999');
            EXECUTE 'INSERT INTO '|| quote_ident(tblName)  ||' SELECT count(*), a.cellcode, ''|| to_timestamp(ts_minutes*60) ||'' FROM ( select anonid, the_geom, ts, cellcode, row_number() over (partition by anonid order by the_geom,ts desc) from '|| quote_ident(relname) ||' where ts <= '|| ts_minutes ||') AS a WHERE row_number = 1
            minutes = minutes + 15;
            ts_minutes = 21758280 + minutes;
            EXIT WHEN ts_minutes = count;
        END LOOP;
        RETURN minutes;
    END

9 http://www.postgresql.org/docs/9.2/static/app-psql.html
10 http://www.pgadmin.org/
```
Example of an UrbanAPI PostgreSQL function

### 3.2 First resulting Data Structures

In order to have a best practice for future users of the UrbanAPI Mobility Explorer application it is necessary to have a well-defined final data structure for the pre-processed and imported data. This structure is (at the moment) based on a 1km grid (finer resolutions seem not to be appropriate due to the relatively coarse location information within the mobile phone location data). Data sets depicting static user density for certain time slices consist of three attribute columns:

- Time slice
- Raster ID and
- User count (as anonymised identity)

where the time-stamp is divided into 15 minute slices for temporal data base queries and their visualisation.

Data sets on interaction data (motion pattern of users between origin and destination raster) have to be extended to 4 columns:

- Time slice
- Origin-Raster ID,
- Destination Raster ID
- User count of users from origin raster entering the destination raster within time slice

This simple, yet powerful data structure allow easy visualisation of the data. The transfer of the content provided through the first data structure into a 3D- environment can be carried out which allows integrating the mobile device density layer into CityServer3D for 3D-visualisation within the urban layout. The transfer of interaction content provided through the second data structure needs an additional translation feature.

### 3.3 Data Export Functionalities

The web-application will be capable of exporting data according to the users’ needs. First the user selects the kind of visualisation way he or she wants to have the result of their request visualised (Figure 8, Figure 9 and Figure 10).
A resulting map (Figure 10) will be displayed and the user will have the possibility (via the “Export” button) to select of the file format for static map or animation export.

The database will be queried by the Mobility Explorer web-application and will view data exploration results according to the users' requests. Regarding the output the user can select between images (i.e. a screenshot), animations (i.e. a video of a selected time range) or tabular data in spreadsheet format, e.g. MS Excel or CSV file format. Additionally it is planned to export content into 3D formats like X3D for reuse results in further applications – e.g. for elaborated 3D presentations.
4 Urban Development Simulator Data

The Urban Development Simulator (UDS) will be developed using MASGISmo a (Multimethod Agent-based, System Dynamics and GIS modelling platform [5]) which has been developed as a basic generic simulation tool at AIT during earlier projects. MASGISmo currently includes a database connection to a PostgreSQL database, where all tabular data are stored.

Geo-data are currently imported and exported through ESRI’s proprietary ASCII raster format. The ASCII grid files are imported during the simulation on the fly.

For the MASGISmo UDS a PostGIS connection was developed, which allows including the support for geographic objects within the PostgreSQL database. Establishing a connection to OpenJump (http://www.openjump.org/) enabled to handle ESRI-shapefiles or other vector based geo-data thus a wider range of data sets can be used for the simulation application without reformatting needs.

MASGISmo, making use of open source software such as PostgreSQL and PostGIS, enabled to adapt to the user’s needs in a wide range of cases. Programming of new interfaces allowed the import and export of new specific data formats.

4.1 Pre-processing of tabular Data

As mentioned above the UDS uses a PostgreSQL database to store tabular data. MASGISmo, the used modelling platform, can already import and export tables stored as CSV-files. These files can easily be extracted from Excel spread sheet files. MASGISmo uses a determined structure for these CSV-files to enable the user to store variable-specific metadata for each column in the DB as a separate table. During data import the meta-information marked as UNIT and COMMENT is automatically stored in separate rows of the data table (see Figure 11). Thus the user can retrieve metadata information for each variable in a table in a convenient way.

Nevertheless standard CSV-files, lacking this function, can also be imported to the DB using an external software as Navicat for PostgreSQL.

---

12 Raster images are visual representations of numerical information such as elevation. They are resolution dependent, this means they cannot scale up to an arbitrary resolution without loss of apparent quality.
15 Vector data defines features by their geometry as a point, line, or polygon. This kind of geographic layer is infinitely scalable without loss of quality.
16 csv...Comma-separated values files
The DB Browser provides functions for viewing and editing data within the table as shown in Figure 12. A modify function will allow simple editing including the integration of SQL statements. The meta information for each column is also presented in the viewer.

![Figure 11: Structure of csv file for data and metadata import](image)

![Figure 12: DB-Browser UI](image)
4.2 Handling of Geo-Data

As described above MASGISmo’s capability was enhanced using not only geodata provided through ASCII grid files but applying also PostGIS content integrating OpenJump. Thus it is possible to use various geo-data file formats including ESRI shapefiles or GML files and import and export GIS layers stored in the connected PostGIS Database.

Figure 13: PostGIS GIS layer UI using OpenJump

Figure 14: Different GIS layer import UI using OpenJump
References

1: J.Rix et al., Grant agreement: Collaborative project Annex I - Description of Work, Project: Interactive Analysis, Simulation and Visualization Tools for Urban Agile Policy Implementation, 2010
2: Thorsten Reitz, Architektur eines interoperablen 3D-GIS unter besonderer Berücksichtigung von Visualisierungsanwendungen, 2005
3: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns – Elements of Reusable Object-Oriented Software, 1995
4: Bertrand Meyer, Object-Oriented Software Construction, 1988
5: http://www.uvmsystems.com/index.php/en/6: A1 data: For a technical description of stored events and record structure etc. see D4.1 Integrated and Harmonised Data
7: Telefonica data: For a technical description of stored events and record structure etc. see D4.1 Integrated and Harmonised Data
8: TIM data: For a technical description of stored events and record structure etc. see D4.1 Integrated and Harmonised Data