

Project number:	288577
Project acronym:	urbanAPI
Project title:	Interactive Analysis, Simulation and Visualisation Tools for Urban Agile Policy
Instrument:	STREP
Call identifier:	FP7-ICT-2011-7
Activity code:	ICT-2011.5.6 ICT Solutions for governance and policy modelling

Start date of Project:	2011-09-01
Duration:	36 month

Deliverable reference number and title (as in Annex 1):	D3.7 Processing Components Documentation
Due date of deliverable (as in Annex 1):	31.8.2013
Actual submission date:	<i>see "History" Table below</i>
Revision:	1

Organisation name of lead contractor for this deliverable:
Fraunhofer IGD

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium	
CO	Confidential, only for members of the consortium (including the Commission Services)	




Title:
Processing Components Document
Author(s)/Organisation(s):
Wolfgang Loibl / AIT, Jan Peters-Anders / AIT, Ernst Gebetsroither / AIT, Michel Krämer / Fraunhofer IGD
Working Group:
WP3 – Fraunhofer IGD, AIT
References:
Description of Work

Short Description:
The report describes the data processing requirements for each of the tools and the processing components applied.
Keywords:
Tools applied for raw data editing, coding – reordering, validation – consistency check, storage

History:				
Version	Author(s)	Status	Comment	Date
001	Wolfgang Loibl	rfc	Initial draft table of contents, intro text	12. 07. 2013
002	Ernst Gebetsroither	rfc	UGS Chapter	31.08.2013
003	Jan Peters Anders	rfc	PME chapter	01.09.2013
004	Wolfgang Loibl	rfc	Minor updates.& changes	02.09.2013
005	Jan Peters Anders	rfc	Minor updates.& changes	17.09.2013

Review:			
Version	Reviewer	Comment	Date
004	Zaheer Khan		

About this Document	4
1 Introduction	4
2 Data processing activities	5
2.1 Data processing and processing components for Application 1 – 3DVR	6
2.2 Data processing and processing components for Application 2 – Public Motion Explorer	6
2.2.1.1 GSM data Vienna:	6
2.2.1.2 GSM data Vitoria-Gasteiz	7
2.2.1.3 GSM data Bologna	9
2.2.2 Coding - reordering	9
2.2.3 Validation – consistency check	9
2.2.4 Storage – data structure ready for application	10
2.3 Data processing and processing components for Application 3 – Urban Growth Simulation	10
2.3.1 Raw data editing	10
2.3.2 Coding – reordering	11
2.3.3 Validation – consistency check	14
2.3.4 Outlook	19
2.3.5 Storage – data structure ready for application	20
3 Annex	21
3.1 Code examples:	21
3.1.1 Variance check function:	21
3.1.2 Time span check function	26

About this Document

This document is deliverable D3.7 of work package 3 of the UrbanAPI project, due in PM24.

The report describes the data processing requirements for each of the tools and the processing components applied.

The document will be superseded by an improved version if required at any time in the project life cycle. The finalized version of this document is planned for month 30

1 Introduction

Urban planning involves various 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, further mobile phone location data and urban land use and statistical data referenced to spatial reference units. This data shall be used to enable an 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 UrbanAPI description of work [1] defines three applications:

- The 3D VR application deals with visualising existing data sets (such as city models, digital terrain models, etc.) as well as simulation results in a scenario-oriented manner.
- The Public Motion Explorer application deals with mobile device data. This data is analysed to information about how citizens move through the city at certain times of the day.
- The urban growth simulation application simulates the effects of structural change in cities.

The UrbanAPI applications will then be used to visualise 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.

The three applications deal with heterogeneous data that has to be integrated and processed in different ways (see deliverable D3.5 “Data integration components”).

Therefore the data processing activities and the required components are different from tool to tool developed within the project.

The next step after processing and harmonizing the data in work package 4 is to integrate it in the UrbanAPI applications.

2 Data processing activities

Data processing refers to the process of converting data from one format to another. It transforms plain data into valuable information and information into data. Data processing services take the raw data and process it accordingly to produce sensible information.

Data processing denotes all actual data manipulating techniques such as classifying, sorting, calculating, summarizing comparing etc. Data processing ensures that the data is presented in a clean and systematic manner and is easy to understand and be used for further purposes.

We list here five steps in data processing¹:

- Editing

There is a great difference between raw data and applicable data. Depending on the application there can be a huge volume of raw data, where useful data has to be extracted. Extracting relevant data is one of the core procedures of data processing to discard the inappropriate data and retain relevant data.

- Coding

After the editing process, the available data is not always in a specific order and has to be aligned into the particular system. The method of coding arranges data in a comprehensible format. The process is also known as netting or bucketing.

- Validation

Data validation refers to the process of thoroughly checking the collected data to ensure optimal quality levels. The entered data are checked for inconsistencies and where possible, they are resolved.

- Tabulation

This is the final step in data processing. The final product i.e. the data is tabulated and arranged in a systematic format so that it can be further applied. Hence appropriate to a topic, further data disaggregation is shown. The first step towards producing the Final Report tables is to generate a "standard recode" data set, which contains the same data as the raw data set, but in a standardized format. It is standardized in that the variable names and definitions are, wherever possible, consistent across all surveys. The "standard recode" is also important for researchers and policy makers since it produces a clean set of data for use. The second step is generating the actual Final Report tables. If possible, a preliminary set of the Final Report tables is generated in the time remaining during this country visit, while the complete set is generated at Macro.

- Storage

¹ <http://www.webproworld.com/webmaster-forum/threads/76231-5-Steps-To-Data-Processing>;
<http://www.measuredhs.com/data/Data-Processing.cfm>

Data processing results are placed in storage to be used as input for further processing – in our case within the 3 tools developed and extended within UrbanAPI. Storage is provided as files - a collection of records, where each record contains similar items. A collection of files results in a data base.

All these processes make up the complete data processing activity which ensures the said data is available for application. The 3 tools deal refer to data sets of different quality - only the public motion explorer deals to some extent with raw data, requiring comprehensive preparation tasks. The 3DVR as well as the Urban growth simulation refer to preprocessed data, unless the latter need additional data correction and gap filling steps.

Data analysis is not summarized below data processing as this is an analytical task taking these preprocessed data as input.

2.1 Data processing and processing components for Application 1 – 3DVR

The data processing was mostly carried out in WP4 using existing standard tools, such as ArcGIS. As CityServer3D already supports a wide range of standard formats, no further special data processing components had to be developed. Only a Converter for the data of the city of Vienna was developed because the data was in a proprietary format of the CityGridDB software. As this is an XML based format this was a relatively simple task of converting the data to CityServer3D's Metamodel. Please see D3.5 for further information on the software architecture of CityServer3D and the processing facilities.

2.2 Data processing and processing components for Application 2 – Public Motion Explorer

As previous deliverables (e.g. D3.5 *Data Integration Components Documentation* or D4.1 *Integrated and Harmonised Data*) have already shown, each mobile provider's data is different from one another and there is no "common" format for these data sets. Hence –from the data processing point of view- there might never be the possibility to integrate a "standard" measure or even a simple import/processing functionality in the PME application. For this reason we will discuss here the steps that had to be undertaken in order to derive a final resulting data set from the delivered "raw" data.

2.2.1.1 GSM data Vienna:

The data delivered by A1 in Austria came in binary log files accompanied with a structural description of the content .

The first raw data parser was first coded Java, the second one in C++, the third one used direct access from MS SQL Server to load the data. The parsed data were transferred into a PostgreSQL database for first data evaluation. Due to performance constraints a temporary migration to MS SQL Server 2012 for faster data manipulation was performed.

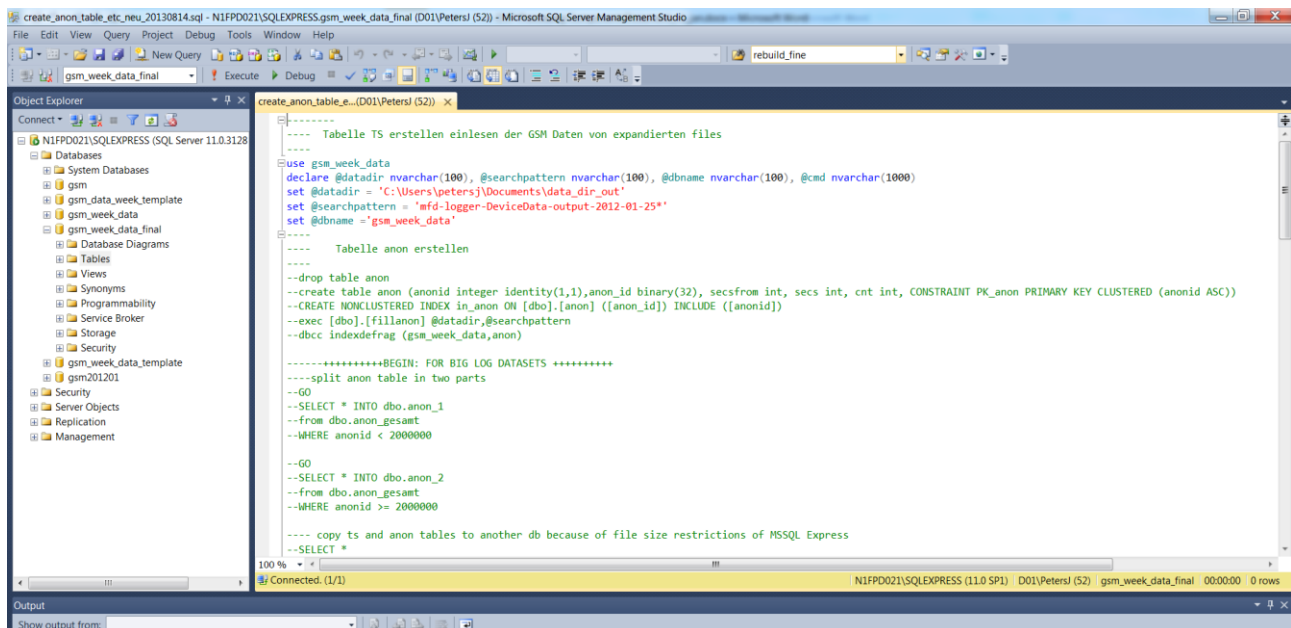


Figure 2-1: MS SQL UrbanApi transactional SQL code

The derived datasets were moved back to the PostgreSQL/PostGIS database for analyses in GIS applications, because of weak GIS analysis capabilities of MS SQL Server. To increase the performance of the SQL functions for extraction of GSM user movements the PostgreSQL functions were developed to handle the huge amount of data faster.

2.2.1.2 GSM data Vitoria-Gasteiz

The GSM data of Vitoria-Gasteiz arrived later in the project. Their data were already compiled in different text files, so they could be processed easily. The files contained e.g the following fields separated by #

- ID_DIA: day id
- ID_USUARIO: user id
- FEC_REGISTRO: date with hours, minutes and seconds
- LLAMADAS: number of calls
- LAC_CI_INI: cell where the call is initiated
- LAC_CI_FIN: cell where the call finishes
- LATITUD_UTM_BTS_FIN: latitude UTM ending cell
- LONGITUD_UTM_BTS_FIN: longitude UTM ending cell
- HUSO_UTM_BTS_FIN: UTM zone, ending cell
- LATITUD_UTM_BTS_INI: latitude UTM initial cell
- LONGITUD_UTM_BTS_INI: longitude UTM initial cell
- HUSO_UTM_BTS_INI: UTM zone initial cell

Accordingly the data was arranged in the following way in the files:

```
16/05/2011#320230007#16/05/2011 19:47:17#1#82#160-9212#160-
9212#4745131#527479#30#4745131#527479#30
16/05/2011#311330007#16/05/2011 19:03:11#2#44#160-42471#160-
42472#4743736#526906#30#4743736#526906#30
```

16/05/2011#399741507#16/05/2011 19:11:12#2#180#102-956#102-953#4744183#526087#30#4744183#526087#30

16/05/2011#399415307#16/05/2011 20:08:59#1#28#160-9281#160-9281#4746391#524815#30#4746391#524815#30

16/05/2011#399291407#16/05/2011 21:46:35#1#301#102-975#102-975#4746536#525480#30#4746536#525480#30

16/05/2011#399263707#16/05/2011 10:14:33#1#41#160-40072#160-40072#4743066#527718#30#4743066#527718#30

16/05/2011#399263707#16/05/2011 11:08:20#1#187#160-9691#160-9691#4744670#527500#30#4744670#527500#30

The text files were processed by transferring them to a PostgreSQL database and the x/y coordinates were transformed into geographic point coordinates. Since the data was containing cell (antenna) based positions only of mobile phone call starts or text message transfers but no mobile phone hand overs, it was only possible to summarize the active users within a certain time slice. Here we have taken the user totals of each cell tower and have calculated totals per cell tower and have average these totals for hourly time slices.

These mean values were then spatially joined and aggregated to 1x1 km raster cells. Within the city centre the size of the raster cells has been reduced to 500x500m to achieve better knowledge about local variation within the centre respectively. This was possible as in the centre the cell tower distribution shows a higher density as in the areas outside the centre. (Figure 2-3).

The processing components used here were PostGIS functions.

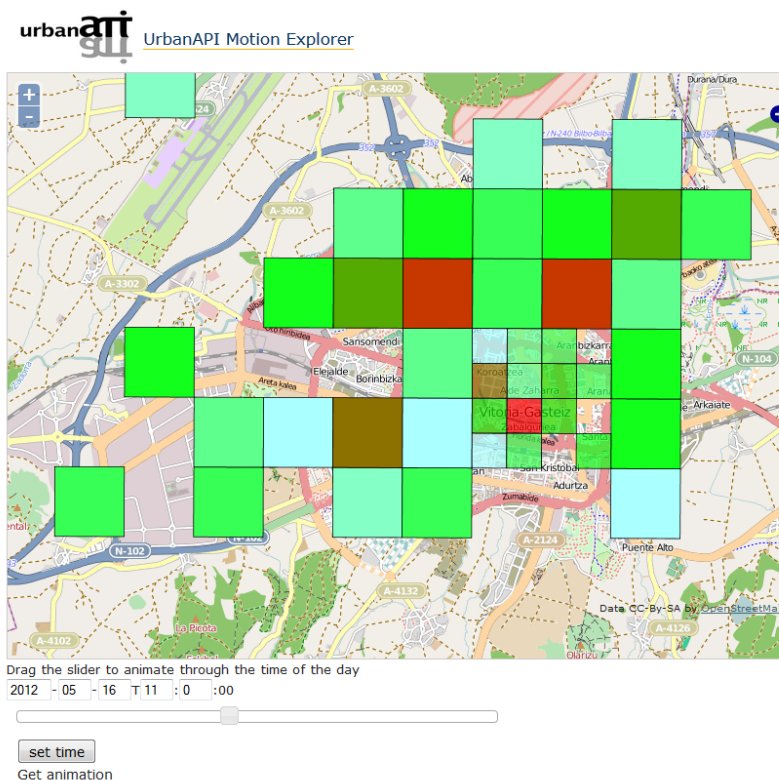


Figure 2-2: Processing results of the Vitoria-Gasteiz data in PME

2.2.1.3 GSM data Bologna

The Bologna GSM data consisted of already aggregated heat maps for certain time steps so there was no further data processing necessary as the data could be directly integrated into the visualisation tool.

2.2.2 Coding - reordering

For all three cities the following processing components (here: applications and extensions) were used:

- Extraction of the raw data: Java, C++ and Transactional SQL in MS SQL Server
- Spatial analysis and calculations: QGIS, PostGIS functions and PostgreSQL as database.

2.2.3 Validation – consistency check

For the Vienna data various checks have been carried out since the data have been delivered as raw data. As a first step the data have been sorted by (anonymous) user ID and timestamp to identify trip chains per user. After this a first check shows some anomalies which turn out to be obvious errors: position changes of single users which would require a speed of up to several 100 km per hour which is not possible – at least in urban environment.

As a second step errors have been corrected by copying the prior valid location position to the later time step.

The third step after error correction builds an analysis via filtering the mobile phone users' locations between 0 and 4 o'clock in the morning and choosing the users' first entry in the log files. This location was defined to be the location where the users are living as assuming the users sleep at home during this time range.

The time span analysis over the day showed in the evening artefact users in the raster cell visualization that did not move back to the location where they came from in the morning. It turned out that the approach to just filter users between 0 and 4 in the morning was too coarse to be used for this analysis because it led to the phenomenon that also users just moving through a certain area at this time (truck and taxi drivers e.g.) were also counted in. This led to the development of a new algorithm which checked the users' travel speed variances (see Annex function 1.).

This function showed an improved picture since it was possible to filter some of the moving users during the time period of 0 to 4 but this was still not satisfying because there still remained artefacts. So a new algorithm was considered:

The next check was to analyse the data set according to the time the users stayed in one place. This algorithm checked the time span a user is staying in one place via checking the location at every new entry in the log. If the user is staying for more than 5 minutes the entry is counted as a "stay" and the location and the user id is transferred to a new result table (see Annex function 2). This is still under investigation but showing better results.

2.2.4 Storage – data structure ready for application

In order to visualize the results the PME works with a final table format which shall later enable future users to transfer their new data into the application. In order to do so they will have to do the pre-processing by themselves since the data sets differ so widely in their formats, but PME will include a manual and examples to enable them to do so. The final table structure looks as follows:

Name	Type	Length	Decimal	Allow Null	
src_cell_id_0_to_4	varchar	0	0	☒	
timestamp	timestamp	6	0	☒	
tgt_cell_id	varchar	0	0	☒	
count	int4	32	0	☒	
the_geom	Type	0	0	☒	
pk	int4	32	0	☐	 1

Table 2-1: Data structure of the PME results table(showing the columns' datatypes)

2.3 Data processing and processing components for Application 3 – Urban Growth Simulation

2.3.1 Raw data editing

Data assembling, generating the GIS base data as well as additional statistical data and preparing them for AIT were done by ASDE 2. Unfortunately most of these data were originally in Cyrillic Bulgarian, so they had to be translated, which was much more complicated as it seemed at the beginning. This translation was done by Geoville. At the same time most of the GIS data had originally no projection information. An attached projection file led to the suspicion that the data might be in this type of projection, but several tries – including different similar projections and slightly changed central meridians, false easting and northing - didn't lead to a satisfactory solution, so our Bulgarian partners had to be contacted to solve this problem. At the end it turned out, that the data were in a special Bulgarian non-standardized projection with no possible transformation in used GIS software (ARC GIS and Quantum GIS) and the attached projection file was wrong. ASDE sent a special tool to transform the original GIS shape files into UTM_zone_35 north projection. Afterwards we could transform them into ETRS_1989_LAEA. All these unexpected problems led to a time delay of almost four month, and so we couldn't start checking the collected data for inconsistencies to ensure their quality and - if possible - to resolve the most serious problems before mid of June!

The GIS data included actual land utilization data of Ruse and Giurgiu, different administrative regions, cadastre plans (including plots and buildings), data about the transport system (e.g. road and railroad networks) and infrastructure (e.g. water supply and sewerage, heating system, gas and power supply) as well as special future plan variants and layers about protected areas, for the Oblast Ruse additionally special

² ASDE Agency of Sustainable Development and Eurointegration – Ecoregions our Bulgarian Projectpartner

statistical sub regions from national statistical bureau NSI and polling region with population data for the last 2 decades. All in all about 180 different shape files (point, line and polygon layers) were available, of course not all useful for the urban growth simulation. On the other side, not all necessary data were available for Ruse-Giurgiu region, so only for a spatial subset - at least for a model prototype - of the whole region including the town of Ruse and the two adjacent *Zemlishte Dolapite* and *Sredna Kula* modeling will be possible (see figure x.1).

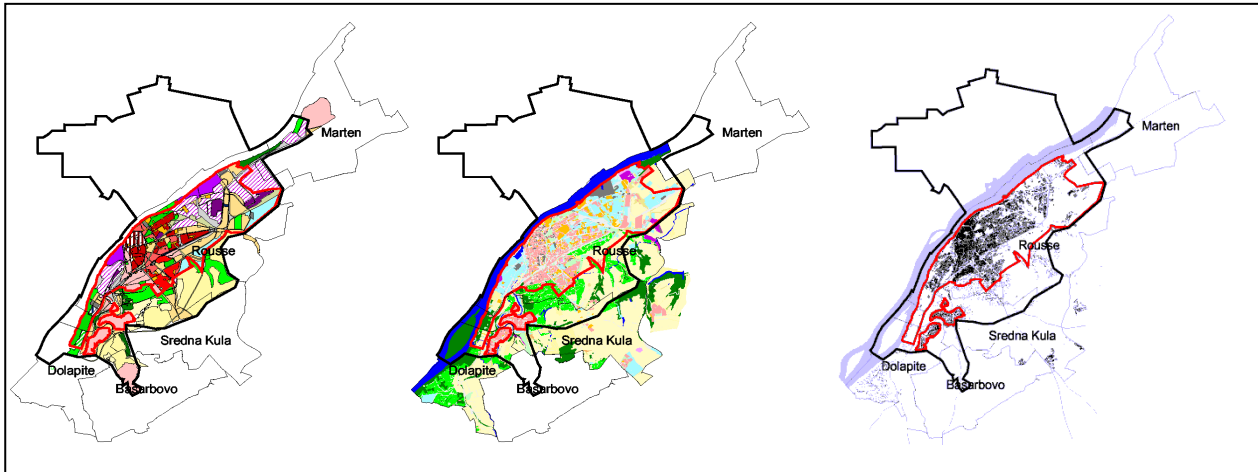


Figure 2-4: zoning plan (left), cadastre with use of plots (middle) and buildings + streets (right); red lines are for the Ruse city border and black

Out of all the different input data we decided to use the following datasets:

- Plots and Building footprints out of the cadastre (including information about height, owner and functional use)
- Zoning plan
- Road and railroad network including stations
- NSI regions, including statistical data about population (age, economic activity and education, family size), buildings (number, age and type), dwellings and households
- Polling-regions for temporal population series
- Manual edited layer with special points of interest for assessing local attractivity (e.g. schools, kindergarten, traffic stations, etc.)

2.3.2 Coding – reordering

For modelling future population development and out of this possible local population pressures leading to demand of new settlement areas and buildings in the Ruse-City region, it is essential to analyze the current situation as well as the immediate past. As said above, for this besides the buildings and plots out of the cadastre statistical NSI regions and polling regions should be used. The source data for the borders of the polling regions were taken from excel files with a list of streets around each polling station and then digitized by ASDE Table x.1 shows an excerpt of this list. The NSI regions came directly from the national statistical institute (NSI), but has been partly digitized by ASDE.

alleys, avenues, places, squares and streets	street number		polling station number №
	odd	even	
Street "Leader Bogdan"	13 - 15 , 23	-	1
Street "Borimechka"	43	40 - 46	1
Street "Voevodova"	43 - 61	36 - 54	1
Street "Vasil Petleshkov"	23 - 39	16 - 28	1
Street "Ivats"	1 - 9	2 -12	1
Street "Major Atanas Uzunov"	-	2 - 22	1
Street "Bridge"	-	4 - 30	1
Street "Hadji Dimitar"	5 , 15 - 19a		
Boulevard "Pridunavski" / Al.Stamb.	-	50 - 66	2
Street "Leader Bogdan"	1 - 7	-	2
Street "Borimechka"	17 - 41	18 - 38	2
Street "Voevodova"	19 - 41	20 - 34	2
Street "Vasil Petleshkov"	9a - 21		
Street "Drach"	1 - 7a		
Street "Ilyu leader"	1 - 11a		
Street "Bridge"	3 - 29	-	2
Street "Stefan Karadza"	1 - 9	2 - 12	2
Street "Han Omortag"	-	2 - 32	2
Street "Chavdar leader"	1 - 33	2 - 30	2

Table 2-2: Excerpt of street boundary list for generating the polling regions

As a result two shape files were generated, because in 2002 the boundaries of the polling stations have changed. Our partners in Bulgaria created only the polling regions with correct description to avoid additional errors and to achieve correct polygons, but this leads to a crucial problem: Several regions were not digitized or identified. So polling region layer 1995-2001 contains 202 polling stations from No.1 to No.218, but the 16 NSI regions of Ruse 103, 155, 177, 192, 194 to 201, 213, 214, 215 and 217 were still missing. The 2002-2012 layer on the other side contains only 168 polling stations from No.1 to No.168. The information about population and voters is taken from DOS software and imported in MS Excel. This table can be joined via section-code with the shape files. Nevertheless the spatial change of the polling region is partly so dramatic (and even the codes show no similarity), that we decided just to consider the region shape file 2002-2012 for the population time series.

Unfortunately the necessary datasets do not cover even the intended subset of Ruse region. Both polling region files - and in fact the NSI region file too – cover a much smaller area. Besides Dolapite and Sredna Kula some other “villages” (e.g. Kadasheva Niva, Doni Rostov and Kaseva Cheshma) are missing. Additionally, compared with NSI regions, the polling regions include for validation year 2011 more population although – as can be seen in figure x.2 – the cover even a less area than the NSI regions. While the 168 polling region table show in year 2011 162765 people and 138797 voters (85.3 %), the NSI show 138903 people. On the other hand in total there are 170416 people in the whole town of Ruse while according to NSI there should be 149642 Inhabitants (<http://www.ruse-bg.eu/en/pages/94/index.html>). It seems that the polling regions also include those people who live actually outside the city, so we decided to use only the annual relative changes of the polling regions standardized by the 2011 NSI population.

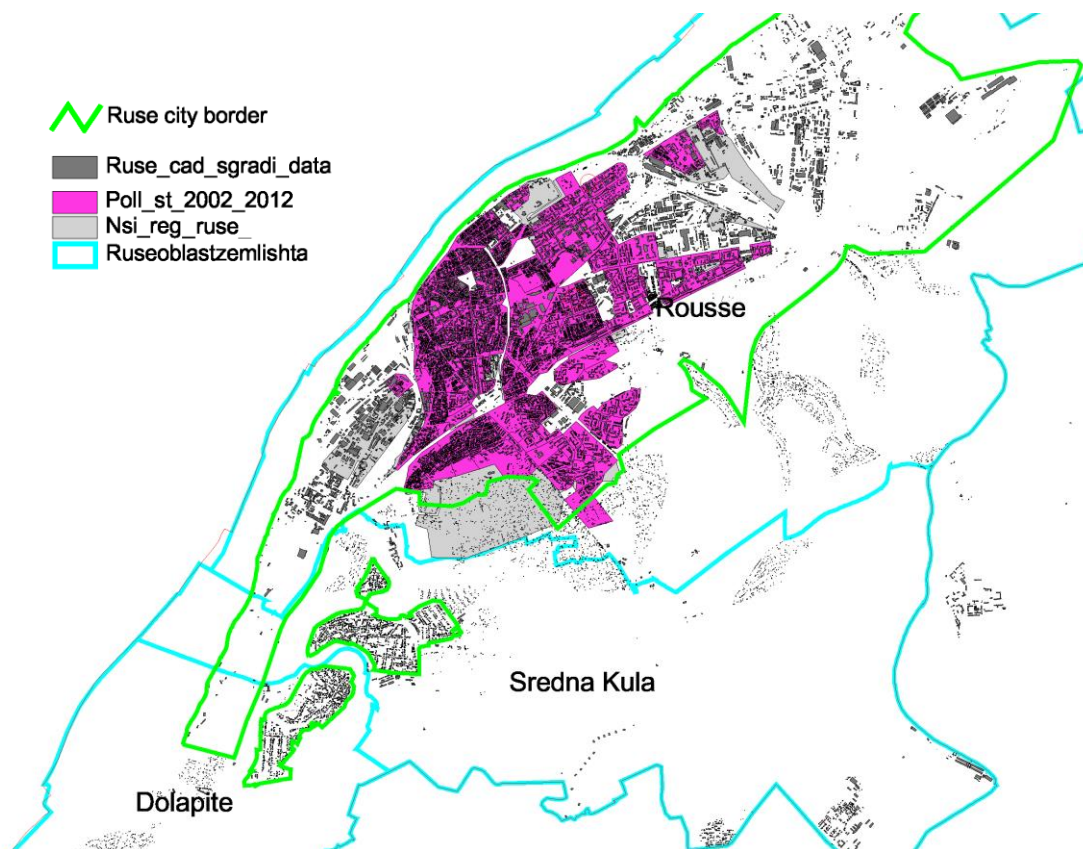


Figure 2-5: Comparison of polling-stations-layer 2002-2012, NSI statistical region layer and buildings of Ruse

But not only in both polling-region layers some regions are missing. Even the NSI layer lacks some regions. So there were no geographical information about NSI region 103 and 194 to 201 including 5650 inhabitants and 4866 buildings. For another 5089 people no region could be identified. While for this amount no building information is available, the statistical data show 4011 households, from which 3685 are single ones leading to an extreme low average household size of 1.14 (while total Ruse has 2.32!).

Several mail-inquiries regarding the missing regions led to the delivery of 7 additional polling regions. Although these regions add up only 5393 people for the year 2011 (~3.6 % of the total population) that's better than nothing. According to information from ASDE the NSI and polling region don't stand in any direct context, but because of our analyses we think that they somehow correspond – the almost identical numeration, but also the similar spatial pattern shown in figure x.3 on the next side seem are a strong evidence for this thesis. A cross-check with the NSI region table confirms this suspicion. While for 2002-2012 the numeration is 170 to 176, the 7 1995-2001 region have the codes 195 to 2001. The 7 NSI regions with these codes include a population of 5039 which is very close to the above mentioned 5393. Anyway, we try to connect the information of the building layer and the NSI regions as well as possible. For all buildings outside the identified regions, we plan to aggregate the remaining population with average statistical data.

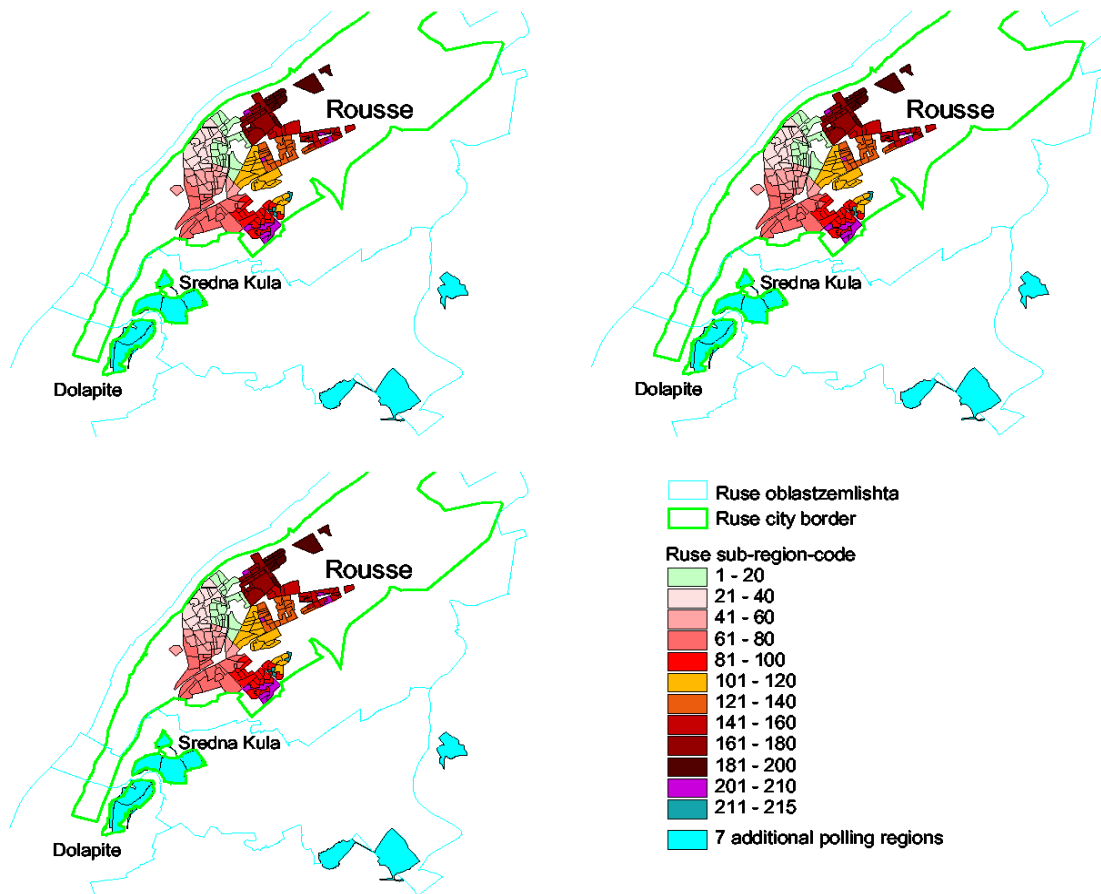


Figure 2-6: Comparison of polling-stations-layer 1995-2001 (upper left) and 2002-2012 (upper right), NSI statistical region layer (left)

2.3.3 Validation – consistency check

Our goal is to combine residential areas with building-information (e.g. height, ownership, etc.), information of the NSI-regions (date 1.2.2011) and polling-data (time-series of population). The following figure shows the planned flow to generate a database for the Urban Growth Simulation. First we plan to disaggregate the input data via the so called “living area” of a building (= area multiplied by the number of floors) of all residential buildings of a special region to single building and afterwards to aggregate the data of the buildings to the 500m regular grid cells. This aggregation to 500m grid cells is an important basis for the Urban Growth Simulation.

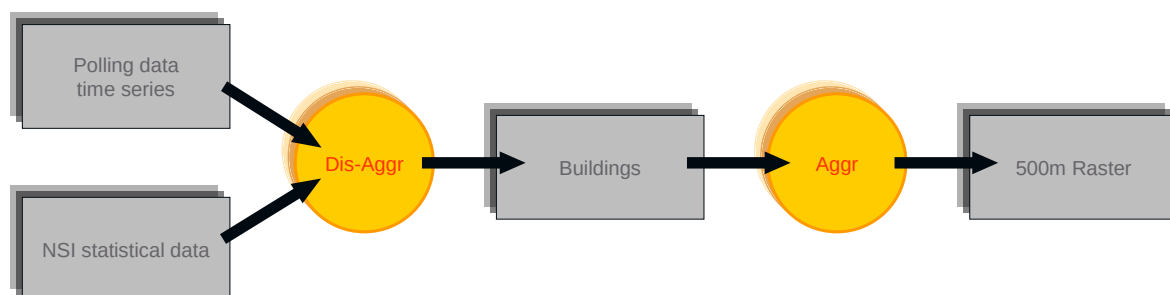


Figure 2-7: Scheme of converting regional data to regular 500m grid

As said above, these GIS-datasets do not match the whole Ruse-City region and much more problematic, some (statistical) sub-regions could not be identified. Nevertheless, all in all an area of about 110000 m² can be considered for the simulation.

Another problem – and maybe the most problematic one - is the selection of those buildings which are used for the dis-aggregation and afterwards aggregation process of the statistical data. We have to find out which buildings are used for living – what is difficult? The cadastre plan we have is not as up to date as the statistical input – some houses might have been pulled down or have changed their shape, others will have been built up. In total the cadastre map of Ruse accounts for 44230 building from which 25716 have residential purpose³, but many of them are in fact schools, kindergartens or administrative and business buildings, and more problematic many of these are big buildings. After several tests we decided to use those buildings within the provided buildings layer with attribute groupcode 1 – “**buildings for permanent and temporary residence**” (see figure below and left list in table x.2). This accounts for 22626 buildings, less than the number (25716) gained by spatial selection with land purpose residential.

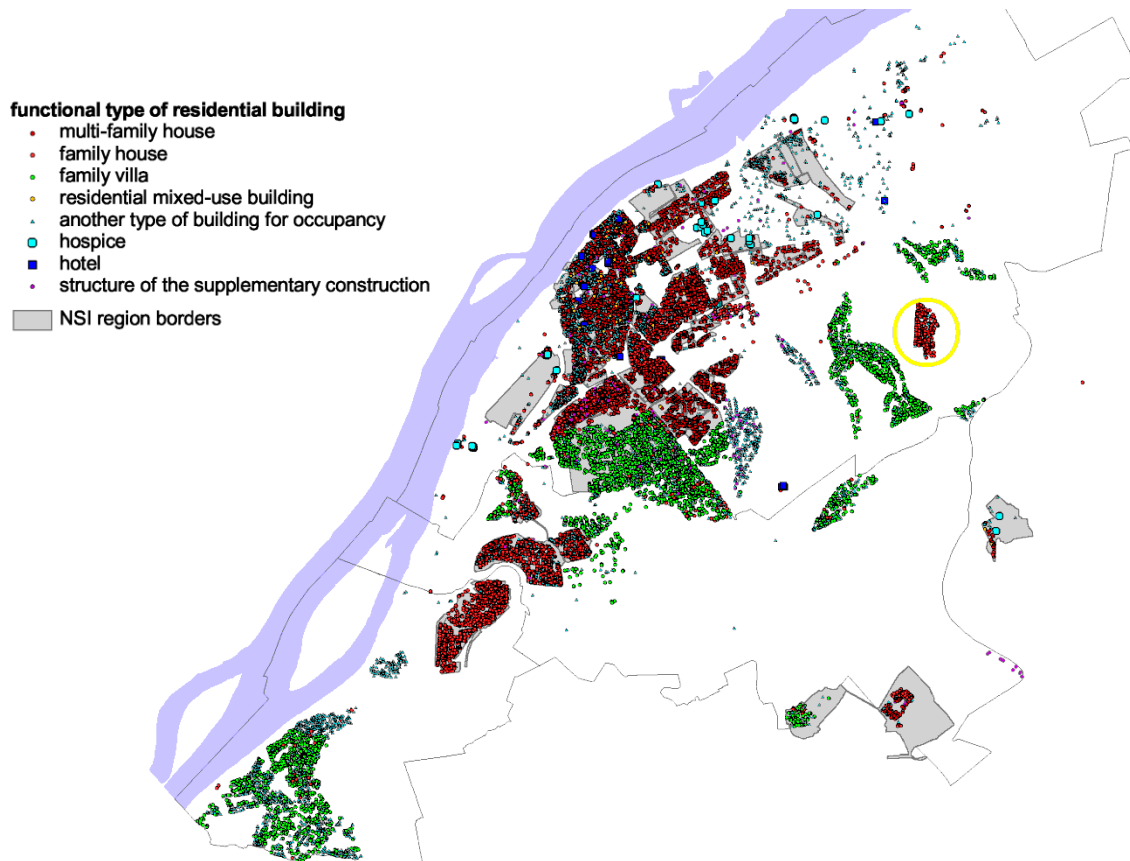


Figure 2-8: function type of 22626 buildings for permanent and temporary residence

This seemed smart, but by looking in detail several new problems occurred as: What means the definition of the building function – e.g. 8832 building (parts) are called “*another type of building for occupancy*”, this are 39% of all total buildings! On the other side: Which buildings (except hotels and maybe hospices) are used

³ extracted by spatial selection of buildings on land purpose residential

for “temporary” residence? Many buildings seem to have a wrong function and sometimes the height is only one floor, although a visual interpretation via Google Maps or other online map providers (e.g. <http://wikimapia.org>) showed, that they are substantially higher. For example: Most of the buildings within the yellow circle in figure x.5 are family houses, whereas the land use purpose says that the whole area is for recreational use. In the end the maybe most demanding question is: How can the type of function within the building layer be compared with the type of the NSI data (right list in table 2-3)? For example: What is a cookhouse and how is the definition of an “apartment block” (height, shape and dimension).

FUNCTION	Nº	NSI Type	Nº
another type of building for occupancy	8832	house	8130
family house	6345	villa	4545
family villa	3883	apartment block	1604
multi-family house	2701	mixed type	100
structure of supplementary construction	669	cookhouse	16
residential mixed-use building	131	hospice	15
hospice	40	collective building household	9
hotel	25	total	14419
total	22626		

Table 2-3: Number and function of “buildings for permanent and temporary residence” out of cadastre (left) and NSI reference (right)

It’s obvious that the total numbers in table 2-3 don’t match. Examining the building layer and comparison with NSI data, we did not use the buildings themselves but much more generated “building blocks”, meaning that all original buildings out from the cadastre are only parts of a greater entity. The following figure 2-9 shows the original “building parts” (left) and the generated “blocks” (right). One has to bear in mind that now building blocks include more and different functional parts, so in many cases it’s not clear what kind of type the block will be. Especially the huge number of the class “another type of building for occupancy” complicates the definition of the type.

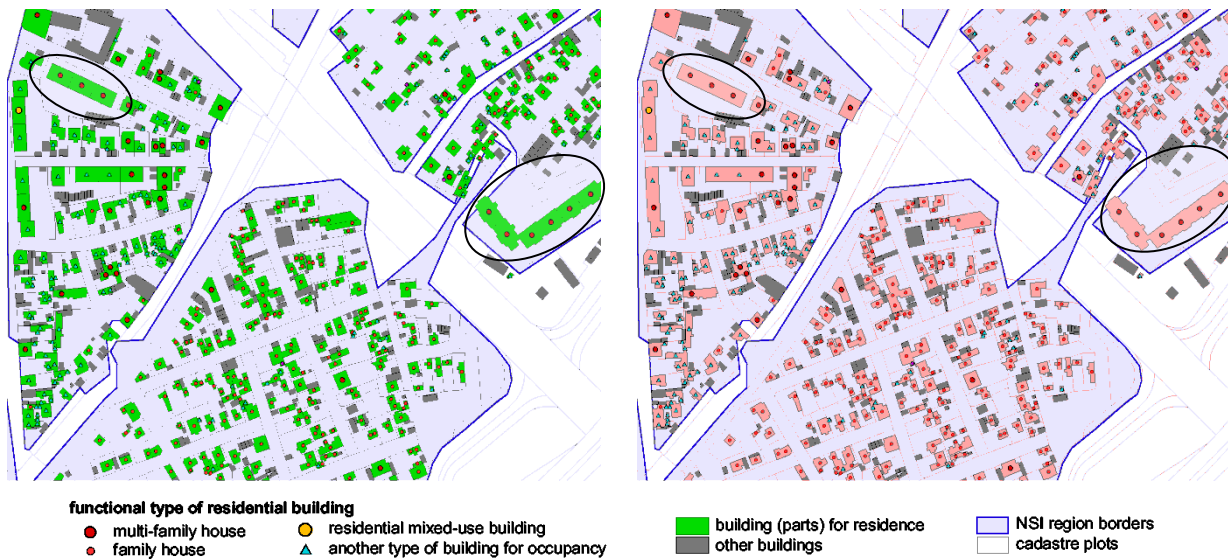


Figure 2-9: original buildings (now called “parts”, left) and generated “building blocks” (right) due to dissolve operation

According to NSI table the statistical regions include 14419 buildings, mainly houses 56.4 % ⁴ and “villas” (31.5 %), for which we assume that they are only for leisure time use and therefore not relevant for dis-aggregation of the population. Most of these so called “villas” are extreme small, thus they should better be called “garden-houses”. This underpins our assumption to leave them out. Additionally hotels are not considered for the distribution. The third important building group are the apartment blocks. They will be most important for population distribution because of their size and/or their height (and therefore living area). Nevertheless we don’t know how an “apartment block” is defined. We defined that such a block must have at least three floors or include more than three housing buildings. If there are different numbers of floors inside such a block it counts for two or more apartment blocks. For building blocks including “residential mixed-use” the living area will be reduced by 30 %. All other types seem cause of their small number not relevant for the validation.

After dissolving the building layer one gets 15679 “building blocks”, which is a pretty good first guess compared to the 14419 “buildings” of the NSI data. The reason for this difference might be that – even after dissolving the parts – many building blocks have extreme small areas – e.g. more than 2300 are smaller than 15 m², 143 of them are pretended to be family houses. Additional 1600 building blocks (111 “family houses”) have an area between 15 and 19 m². Obviously these buildings are mostly garages or – if the lie within recreational areas – the so-called “villas”. The smallest building in the dissolved building block layer with the function “family house” has only 2.6 m², the smallest object at all has even only 1.7 m². Objects smaller than ~8 m² (492!!) are probably not even garages. Anyway, garages, sheds or storage buildings aren’t part of the NSI building counting.

Nevertheless, at the end the criteria has to be depended on the actual NSI region to get a satisfactory consistency within it and to be sure that the dis-aggregation of the population – and depending on that household-size and other statistical parameters – can be with the necessary accuracy. Table x.3 shows an

⁴ but no differentiation if single or multi-family houses, so if there are more than one family house inside a block, the block still counts for one “house” according to NSI definition)

excerpt of a parameter list for this examination.

control rayoni	type of building (NSI)								estimation via building blocks										poptotal	living area	liv.ar p cap	liv. Ar p cap15+	liv.ar p hh	liv.ar p dwelling
	total	house	apartment block	mixed type	hospice	cook house	villa	collective building household	total	house	apartment block	mixed use	hospice	villa	other									
total Ruse	14419	8130	1604	100	15	16	4545	9	14498	8294	1533	104	20	4547	20	149642	5837254	39.0	44.4	91.1	79.6			
Ruse 001	89	67	20	1	1				89	68	20	1				806	41714	51.8	59.2	114.6	90.9			
Ruse 002	205	187	14	3		1			206	188	14	4				792	47145	59.5	67.3	136.3	93.7			
Ruse 003	3		3						3		3					463	25511	55.1	61.9	122.1	109.0			
Ruse 004	140	117	23						141	118	23					699	38639	55.3	62.9	127.5	86.8			
Ruse 005	24	13	11						23	12	11					416	20284	48.8	54.1	111.5	95.7			
Ruse 006	46	32	14						46	35	11					461	33344	72.3	82.7	153.0	111.9			
Ruse 007	150	109	29	12					152	115	29	8				1326	88838	67.0	80.7	159.8	111.5			
Ruse 008	184	148	20	10		6			184	163	18	3				1100	45977	41.8	49.9	100.6	77.8			
Ruse 009	45	36	7	2					46	37	7	1		1		348	21271	61.1	73.6	153.0	125.9			
Ruse 010	170	163	7						170	164	6					717	30004	41.8	48.5	106.0	88.8			
Ruse 011	98	78	19	1					98	79	17	2				604	33442	55.4	63.9	133.2	90.1			
Ruse 012	1		1						1		1					548	24380	44.5	50.2	103.7	91.3			
Ruse 013	13	5	8						12	5	7					422	22503	53.3	59.1	116.6	95.8			
Ruse 014	14		14						12		12					592	25330	42.8	47.8	104.2	97.4			
Ruse 015	32	16	15	1					24	11	11	2				576	24768	43.0	48.8	99.5	87.2			
Ruse 016	7	3	4						10	5	5					520	42251	81.3	89.3	179.0	150.9			
Ruse 017	50	30	18	2					49	29	18	2				587	32883	56.0	62.3	123.6	103.7			
Ruse 018	35	15	20						34	14	20					450	31828	70.7	81.6	161.6	130.4			
Ruse 019	35	14	15	6					34	18	14	2				713	35871	50.3	56.8	106.4	79.7			
Ruse 020	30	8	18	4					29	9	17	3				764	66471	87.0	97.3	185.7	132.7			
Ruse 021	17	1	10	6					13	1	12					520	27334	52.6	58.3	119.4	100.1			
Ruse 022	11	1	10						11	1	9	1				471	25537	54.2	60.8	127.7	104.7			
Ruse 023	71	45	22	4					70	46	22	2				482	35686	74.0	81.3	153.2	117.0			
Ruse 024	36	26	10						36	27	9					273	18579	68.1	75.8	149.8	116.1			
Ruse 025	56	35	20	1					56	35	18	3				678	32207	47.5	53.9	105.6	79.1			
Ruse 026	119	94	24	1					120	94	22	4				650	44947	69.1	77.5	158.8	119.5			
Ruse 027	90	68	22						91	69	21	1				583	47403	81.3	90.6	178.2	132.0			
Ruse 028	36	27	9						36	27	9					609	30767	50.5	57.2	116.1	92.4			
Ruse 029	99	69	25	3		2			98	70	28					551	38724	70.3	80.3	158.1	120.3			
Ruse 030	29	7	21	1					31	10	21					669	44324	66.3	73.5	141.6	127.4			
Ruse 031	56	37	19						56	37	16	3				703	53610	76.3	84.7	175.8	131.1			
Ruse 032	39	19	10	10					44	26	16	2				378	29331	77.6	86.3	148.1	119.7			

Table 2-4: Excerpt of a table for comparison of NSI data and estimation – green highlighted are matching with the NSI data (table work in progress in the end all cells should be green-clarified)

As one can see, in most of the NSI regions a satisfactory consistency was achieved in the first iteration. Only in a few regions (yellow to red) higher deviations occur. These regions had to be analysed in a second step. Doing this, it was found out, that in some cases the demarcation of the NSI regions could not be right: the difference of the number of buildings was to extreme. It was necessary to adapt and extend some regions manually (see figure x.7 below). Additionally – prematurely near region borders in the city - the automatically joined NSI codes of the building blocks had to be changed to other codes to get consistency.

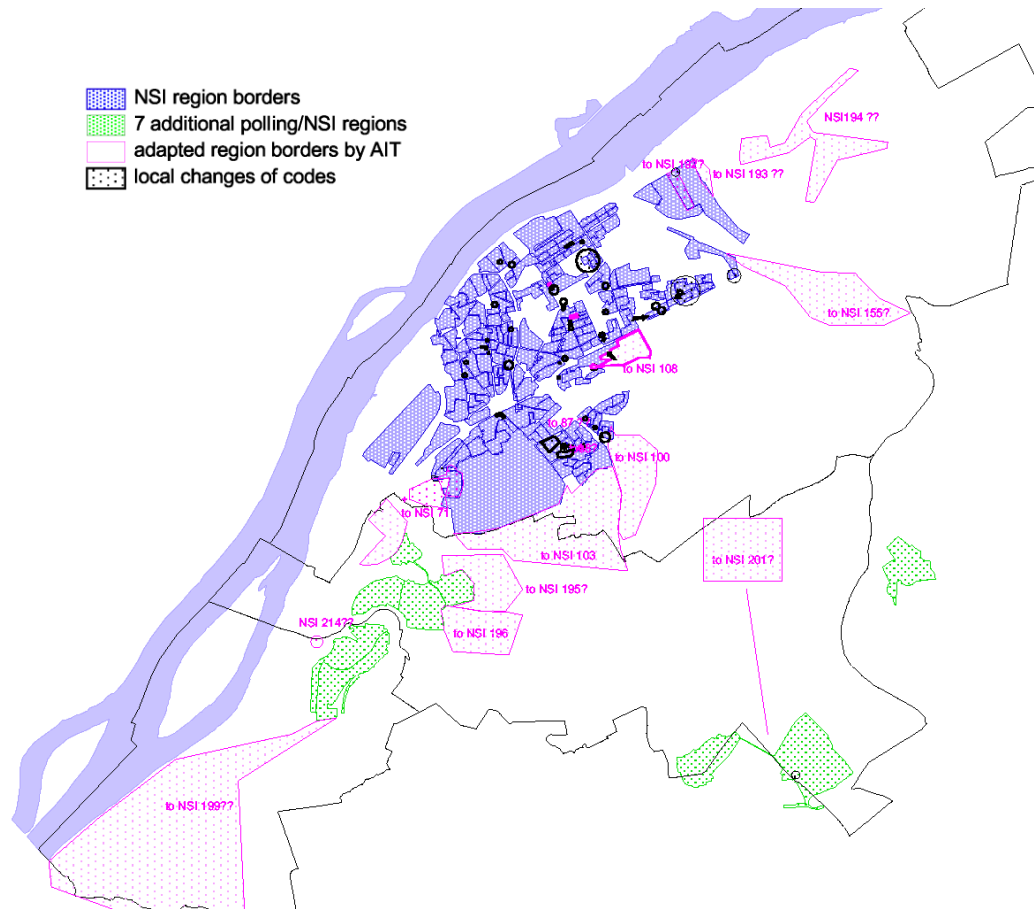


Figure 2-10: original NSI regions (blue and green) and manual adaptation in magenta

Only in very few cases additional new buildings had to be digitized (especially if the buildings are huge and high), or the number of floors had to be corrected. Because it seems that in the cadastre new or just about to be built buildings have only one floor and a comparison with online mapping services showed that this is not the case anymore. Considering these changes of building structure and the temporal difference between building layer and statistical data, the consistency is surprisingly good. With all the applied steps disaggregation of the statistical data will lead to a data set with valuable information. Furthermore the following aggregation to 500m grid-cells will too extinguish some minor inconsistencies.

2.3.4 Outlook

The data validation procedures described in the section above shows that, although some difficulties occurred we at least could establish to get a basic dataset for the first Urban Growth Simulation (UGS) prototype. Nevertheless it has to be noted that the data situation is not as convenient as we assumed, but we already have been aware of these possible problems at an earlier stage of the project. Because of this we decided to build our UGS flexible enough to include better data if it is available in the future with a manageable amount of effort for data pre-processing.

2.3.5 Storage – data structure ready for application

The pre-processed and validated data will be stored in several formats, on the one hand in common GIS files as ESRI-shapefiles, Excel tables and on the other hand within the PostgreSQL/PostGIS database of the UGS model.

3 Annex

3.1 Code examples:

3.1.1 Variance check function:

```
CREATE OR REPLACE FUNCTION "geodb_pkg"."check_timeframe_users_movement_variance"(cellcode
varchar, log_date varchar, time_from varchar, time_to varchar)
  RETURNS "pg_catalog"."text" AS $BODY$--last modified: 28.03.2013
DECLARE
  temp_cnt int4;
  text varchar;
  relname_tst varchar;
  relname_ttst varchar;
  query text;
  result text;
  starttime time;
  stoptime time;
  timeresult time;
  ts1 int4;
  ts2 int4;

  tsp1 timestamp;
  tsp2 timestamp;

BEGIN
temp_cnt := 0;
relname_tst = cellcode|| '_tst';
relname_ttst = cellcode || '_ttst';
result = temp_cnt;
starttime := CURRENT_TIME(1);
SELECT to_timestamp(time_from, 'YYYY-MM-DD HH24:mi:ss') at time zone INTERVAL '+01:00'
INTO tsp1;
SELECT to_timestamp(time_to, 'YYYY-MM-DD HH24:mi:ss') at time zone INTERVAL '+01:00'
INTO tsp2;
--tsp1 = to_timestamp(time_from, 'YYYY-MM-DD HH24:mi:ss');
SELECT EXTRACT(EPOCH FROM tsp1 )/60 INTO ts1;
SELECT EXTRACT(EPOCH FROM tsp2 )/60 INTO ts2;
```

```

-----snip 1 begin-----
--get the earliest entry of each user before 6 o'clock (the earliest entry for each
device, device/user is unique)
BEGIN

RAISE INFO 'tsp1: % tsp2: %' , tsp1, tsp2;
--RAISE INFO 'starting night % %', tsp1, tsp2;

EXECUTE '
DROP TABLE IF EXISTS temp._'|| log_date ||'_variance_check CASCADE;
CREATE TABLE temp._'|| log_date ||'_variance_check AS
select anonid,positionid,ts, the_geom, row_number()
over (partition by anonid order by ts, positionid)
from tts_'|| log_date ||'_3035
where ts > '||ts1||' AND ts <= '||ts2||';
';
RAISE INFO 'night finished.';
END;

BEGIN
EXECUTE '
DROP TABLE IF EXISTS temp._'|| log_date ||'_variance_check_cell CASCADE;
CREATE TABLE temp._'|| log_date ||'_variance_check_cell AS
SELECT a.*
FROM temp._'|| log_date ||'_variance_check a, "temp".temp_raster_cell_'||cellcode||' b
WHERE ST_Contains( b.the_geom, a.the_geom);
';
RAISE INFO 'night finished.';
END;

BEGIN
EXECUTE '
DROP TABLE IF EXISTS temp._'|| log_date ||'_variance_check_cell_ranked CASCADE;
CREATE TABLE temp._'|| log_date ||'_variance_check_cell_ranked AS
SELECT anonid,positionid,ts,
rank() OVER (PARTITION BY anonid
ORDER BY "ts" ASC)
FROM temp._'|| log_date ||'_variance_check_cell;

```

```

';
RAISE INFO 'night finished.';
END;

BEGIN
EXECUTE '
DROP TABLE IF EXISTS temp._'|| log_date ||'_variance_check_cell_ranked_the_geom CASCADE;
CREATE TABLE temp._'|| log_date ||'_variance_check_cell_ranked_the_geom AS
SELECT a.anonid, a.positionid, a.ts, a.rank, b.the_geom
FROM temp._'|| log_date ||'_variance_check_cell_ranked a
INNER JOIN geodb_pkg.positions_'|| log_date ||'_3035 b ON a.positionid = b.positionid
';
RAISE INFO 'night finished.';
END;

BEGIN
EXECUTE '
DROP TABLE IF EXISTS temp._variance_check_cell_ranked_the_geom_segs_stat CASCADE;
CREATE TABLE temp._variance_check_cell_ranked_the_geom_segs_stat AS
SELECT a.ts AS ts_1, b.ts AS ts_2, a.anonid, a."rank" AS rank_1, b."rank" AS
rank_2,a.positionid AS position_1, b.positionid AS position_2,/*a.event_type AS event_1,
b.event_type AS event_2,*/
ST_MakeLine(a.the_geom, b.the_geom) AS the_geom,
ST_AsText(a.the_geom) AS geom_1, ST_AsText(b.the_geom) AS geom_2
--a.pk, ST_MakeLine(a.the_geom, b.the_geom) AS the_geom
FROM temp._'|| log_date ||'_variance_check_cell_ranked_the_geom a,
      temp._'|| log_date ||'_variance_check_cell_ranked_the_geom b
WHERE a.rank+1 = b.rank AND a.anonid = b.anonid
ORDER BY a.anonid;
';
RAISE INFO 'night finished.';
END;

BEGIN
EXECUTE '
ALTER TABLE temp._variance_check_cell_ranked_the_geom_segs_stat
ADD COLUMN duration_h float4;

ALTER TABLE temp._variance_check_cell_ranked_the_geom_segs_stat
ADD COLUMN length_km float4;

```



```
ALTER TABLE temp._variance_check_cell_ranked_the_geom_segs_stat
ADD COLUMN travel_speed_kmh float4;
```

```
ALTER TABLE temp._variance_check_cell_ranked_the_geom_segs_stat
ADD COLUMN variance float4;
```

```
';
RAISE INFO 'ALTER TABLE finished.';
END;
```

```
BEGIN
EXECUTE '
update temp._variance_check_cell_ranked_the_geom_segs_stat
--SET duration_h = EXTRACT (EPOCH FROM ts_2 - ts_1)::float4/3600;
SET duration_h = (ts_2 - ts_1)::float4/60;
';
RAISE INFO 'UPDATE duration finished.';
END;
```

```
BEGIN
EXECUTE '
update temp._variance_check_cell_ranked_the_geom_segs_stat
set length_km = st_length(the_geom)/1000;
';
RAISE INFO 'UPDATE length finished.';
END;
```

```
BEGIN
EXECUTE '
update temp._variance_check_cell_ranked_the_geom_segs_stat
set travel_speed_kmh = length_km/duration_h;
';
RAISE INFO 'UPDATE speed finished.';
END;
```

```
BEGIN
EXECUTE '
update temp._variance_check_cell_ranked_the_geom_segs_stat a
set variance = check_variance(a.anonid, '', '');

```

```

';
RAISE INFO 'UPDATE speed finished.';
END;

--*/--snip2 end-----
BEGIN
EXECUTE '
DROP TABLE temp._'|| log_date ||'_variance_check;
DROP TABLE temp._'|| log_date ||'_variance_check_cell;
DROP TABLE temp._'|| log_date ||'_variance_check_cell_ranked;
DROP TABLE temp._'|| log_date ||'_variance_check_cell_ranked_the_geom;
';

stoptime = CURRENT_TIME(1);
timeresult = stoptime - starttime;

RAISE NOTICE 'Time ellapsed: %', timeresult;

--EXECUTE 'select count(*) AS cnt from _'|| log_date ||'_users_night_'|| cellcode
||'_geom;' INTO result;
END;

temp_cnt := temp_cnt +1;
RETURN result ;
END

$BODY$
  LANGUAGE 'plpgsql' VOLATILE COST 100
;

ALTER FUNCTION "geodb_pkg"."check_timeframe_users_movement_variance"(cellcode varchar,
log_date varchar, time_from varchar, time_to varchar) OWNER TO "postgres";

```

3.1.2 Time span check function

```

CREATE OR REPLACE FUNCTION
"timecheck_positions"."get_real_positions_of_gsm_users_with_timecheck"(tablename varchar,
timespan_minutes int4, anonid int4)
  RETURNS SETOF "timecheck_positions"."gsmfilteredtsreturntype" AS $BODY$
DECLARE

    sql text;
    dummy text;
BEGIN
    --IF lower($1) = lower(YEAR_CONST) THEN
    --    select cast(cast(incrementvalue as character varying) || ' year' as interval)
into intervals;
    -- ELSEIF lower($1) = lower(MONTH_CONST) THEN
    --    select cast(cast(incrementvalue as character varying) || ' months' as interval)
into intervals;
    -- END IF;
    -- dateTemp:= inputdate + intervals;

sql = 'SELECT * FROM(
SELECT sub_a.anonid, sub_a.positionid, sub_b.ts - sub_a.ts as diff_ts,
sub_a.ts,sub_a.cnt,sub_a.rownr, sub_a.rankal FROM(
select x.anonid,x.positionid, x.ts, (x.ts - y.ts) as diff, x.cnt, x.rownr, x.rankal,
y.ranka2 FROM(
SELECT anonid,positionid, ts, cnt, rownr, rank() OVER (PARTITION BY anonid ORDER BY ts
ASC) as rankal
FROM(
SELECT * FROM timecheck_positions.get_real_positions_of_gsm_users(''||tablename
||'',0,'||anonid ||')
rank(anonid)) sub1
WHERE anonid = '||anonid ||') x,
(SELECT anonid,positionid, ts, cnt, rownr, rank() OVER (PARTITION BY anonid ORDER BY ts
ASC) as ranka2
FROM(
SELECT * FROM timecheck_positions.get_real_positions_of_gsm_users(''||tablename
||'',0,'||anonid ||')
rank(anonid)) sub2
WHERE anonid = '||anonid ||') y
where x.rankal + 1 = y.ranka2) sub_a,

(
select x.anonid,x.positionid, x.ts, (y.ts - x.ts) as diff, x.cnt, x.rownr, x.rankb1,
y.rankb2 FROM(
SELECT anonid,positionid, ts, cnt, rownr, rank() OVER (PARTITION BY anonid ORDER BY ts
ASC) as rankb1
FROM(
SELECT * FROM timecheck_positions.get_real_positions_of_gsm_users(''||tablename
||'',0,'||anonid ||')
rank(anonid)) sub3
WHERE anonid = '||anonid ||') x,

```

```

(SELECT anonid,positionid, ts,cnt, rownr, rank() OVER (PARTITION BY anonid ORDER BY ts
ASC) as rankb2
FROM(
SELECT * FROM timecheck_positions.get_real_positions_of_gsm_users(''||tablename
||'',0,'||anonid ||')
rank(anonid)) sub4
WHERE anonid = '||anonid ||') y
where x.rankb1 + 1 = y.rankb2) sub_b
where sub_a.ranka1 + 1 = sub_b.rankb1) as subc
WHERE subc.diff_ts > '||timespan_minutes ||';
';

RETURN query execute sql;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE COST 100
ROWS 1000
;
;
```