Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria Geomatica e dei Trasporti

Ciclo XXV°

Settore Concorsuale di afferenza: 08/A3

Settore Scientifico disciplinare: ICAR 05

Automated Analysis Methods for the Assessment of Bicycle Infrastructure and Demand

Presentata da: Dott. Ing. Pashkevich Anton

Coordinatore Dottorato

Chiar.mo Prof. Ing. Bitelli Gabriele

Relatore

Chiar.mo Dott. Ing. Schweizer Joerg

Esame finale anno 2013

Table of Contents

Table of Figures

Table of Abbreviations

OSM – OpenStreetMap

WP – work package (consequently, WP3 – work package 3 and so on)

PP – Project Partner

CE – CENTRAL EUROPE

LP – Lead Partner

SC – Steering Committee

UniBO – University of Bologna

DICAM

EU – European Union

MA - Managing Authority

JTS – Joint Technical Secretariat

CROW – The National Information and Technology Centre for Transport and Infrastructure in the Netherlands

Introduction

Transport sector was and is one of the fundamental parts of peoples' everyday life for a long time. But only during the last decades it brought a great number of problems including increase of demand for movement, growth of motorization level, increase of private car usage, efficiency impairing of public transport, congestions and air pollution. Many of the named problems are closely connected to each other and require weighed common solutions. That is why countries and the whole regions start to pay great attention to the development of a sustainable mobility and as a result to the policy of the transport sphere.

One of the possible decisions for the above-listed negative aspects is promotion and support of bicycle usage as an alternative for individual transport. Some steps and actions have already been done and are being done today not only by the governments of different countries, but also by local cities and municipalities in different regions. In the EU, especially, were done a lot in the direction of cycling development at all levels:

- A number of appropriate principles was decelerated as part of the goals or as part of the approaches for achievement of these goals in the European Union policies and White Paper of European transport strategy.
- There is an amount of EU foundations which supports different projects on the regional and country levels.
- Also local programs are carried out to promote and to increase a bicycle usage in a concert place.

These theses are closely connected with one of the European regional project named the BICY project. That is why in the first chapter this project will be considered in details: its main goals, role, connections with the common European Strategy will be described; also the project structure and all processes flow will be explained. In the end one of the BICY project part – Work Package 3 (WP3) – and its component will be considered and discussed in details as the main focus of these doctoral work. Here not only connections between WP3 activities and theses will be considered, but also the sense of theses topic and its development directions will be cleared. An explanation will be given how named action of the BICY project helped to realize and to develop the main ideas of this doctoral research. Finally, also a role of activity “Analysis of Good Practices” for theses will be presented.

Chapter 2 will be devoted to one element of the BICY project which was mentioned in the end of the previous chapter – Analysis of Good Practices. Here the following subjects will be studied: history of cycling in Europe, cycling mobility in the Netherland and its cities compared with other countries, Dutch experience in case of bicycle-friendly infrastructure planning. In the end of chapter 2 the project Cycle Balance will be described in details. It is every good example which presents a complex study of cycling mobility and cycling conditions in different places in the Netherlands. It can be considered as preparation phase for research done within the scope of theses. Also it returned a number of orientating points which helped to cover the main topic of doctoral dissertation and to reach its main goals.

Automated analysis methods for the assessment of bicycle infrastructure and demand, their creation and development are the main aims of this doctoral dissertation. These two methods will be considered in details in chapters 3 and 4.

The first of two automated analysis methods will be considered in chapter 3. Chapter 4 will describe the second method in details.

Chapter 1 – BICY Project

1. Main courses of European Transport Strategies

As was written above, the EU has its main transport strategy documents where development of cycling mobility was marked directly and indirectly as one of the priorities for the future. But the real growth of interest to this issue can be seen only by comparing two latest White Papers (of the year 2001 and of the 2011). Table 1.1 presents the main goals of these two fundamental European transport strategy documents.

Table 1.1 – The main goals of White Papers

Main goals of White Paper 2001	Main goals of White Paper 2011
• Introducing new regulations • Revitalising railways • Optimal usage of infrastructure • Linking up the modes of transport	• Reducing emission • Multimodal urban transport • Efficient usage of exicting infrastructure

According to this table, the most named topics of both Papers are very similar or very close to each other. For example, The White Paper 2001 was promoted to make railway usage possible for urban and suburban areas which can be defined as the first step for creation of multimodal urban transport system – one of the White Paper 2011's aims. Another example is connected with organization of cost-effective transport network by establishing better connection between inland waterways, sea and rail transport which was described in the earlier document. Realization of this aim helped to reach an optimality in the infrastructure usage which was a precondition for usage of improved traffic management and information systems. Utilization of these systems is provided by the White Paper 2011.

The total new aspect of White Paper 2011 in comparison with White Paper 2001 is a reducing CO2 emission. To be exact, EC wants to reduce emission by 60%. Of course, other goals of strategy have direct and indirect positive influences on this point (especially, development of traffic management systems can be mentioned). But to reach this concrete aim some special priorities were defined:

- new transport patterns must emerge according to which large volumes of freight and great numbers of passengers are carried to their destination by the most efficient modes;
- individual transport must be preferably used only for the final kilometers of the journey;
- clean vehicles must be developed and utilized;
- bicycle must become more popular as a separate mode of transport

The last priority connected with growth of the bicycle usage has not only one outcome as decrease of CO2 emission, because bicycle as a mode of transport does not generate any exhaust gases. It also keeps the people healthy and gives a possibility for urban mobility not to depend on other transport means which has a positive effect on an urban transport sector.

For increasing a cycling rate, a number of actions must be provided according to the White Paper 2011. For example, one of them is to support bicycle by other modes of transport: creation of possibilities and backgrounds for taking bike on the bus, train, tram or whatever public transport. It will extend the opportunities to use bicycle on the connections with short distances which are normally done on foot or with other modes of transport (between bus stops and travel destinations, etc.). Other example is a development of a cycle infrastructure or modernization of existing transport network to make it more bicycle-friendly: building of cycle ways, adaptation of some roads for bicycle usage with help of special traffic signs, etc.

2. BICY – Cities and Regions of Bicycles

There is a number of approaches used by EU government to support and to realize the ideas and priorities established in the strategy and policy transport documents. One of these approaches is to organize and to finance an amount of different programs and projects in Europe which activities will be aimed at declared goals.

A sample of such programs is the European Territorial Co-operation which is financed by the European Regional Development Fund and includes cross-border, transnational and interregional co-operation programmes. On the transnational level it divides into 13 different parts. Each part corresponds to one of the transnational cooperation areas. The largest one between these areas is Central Europe which is directly connected with the CENTRAL EUROPE Programme.

CENTRAL EUROPE is an EU programme which supports the cooperation between the countries of Central Europe. During the Structural Funds Period from 2007 until 2013 it encouraged projects which goals met one of the following priorities:

1. Facilitating Innovation across Central Europe

2. Improving Accessibility of and within Central Europe
3. Using our Environment Responsibly
4. Enhancing Competitiveness and Attractiveness of Cities and Regions

"BICY – Cities and Regions of Bicycles" was one of the projects approved by CENTRAL EUROPE. The short name of this project is BICY which was reflected in the logo (Figure 1.1).

Figure 1.1 – BICY logo (source bicy.it)

The official duration time was from February 2010 until January 2013. During this time 11 partners from 10 regions of 7 different Central Europe countries participated in collaboration activities (Figure 1.2).

Figure 1.2 – Partner cities on the BICY project (Source: bicy.it)

The common goals of the BICY corresponded to the second priority declared by CENTRAL EUROPE and, specifically, to its part connecting with promoting a sustainable and safe mobility. By looking more in details, the project wanted to make more people use bikes or walking as a mode of transport trying in such way to change the trend of private motorization increasing in Central Europe and, as a result, to reduce all its negative effects inclusive of traffic congestion and air pollution.

Considering that 80% of all trips in urban areas are less than 5 km, it is natural that walking and cycling would be the best modes of transport: they do not generate any CO2, improves people's mobility without depending on any other modes of transport and promotes a healthy life-style. Figure 1.3 shows the advantage of bicycle for the short-distance trips versus other modes of transport. But this still does not mean that public transport should become extinct, instead of that, walking and cycling should be integrated into it.

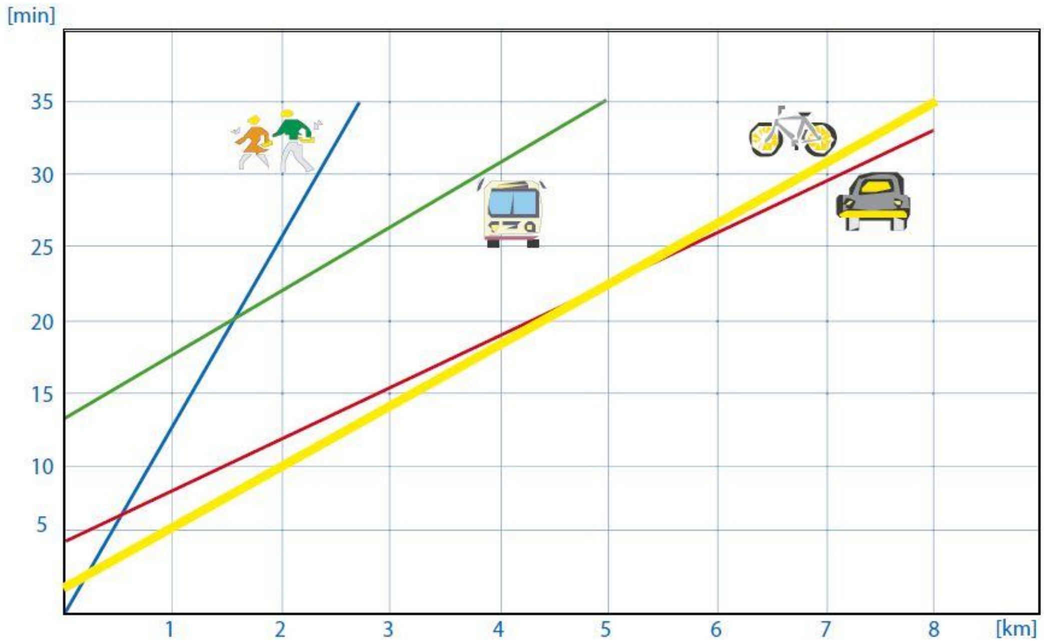


Figure 1.3 – Short-distance trips by different modes of transport (Source: Trendy Cycling)

So in particular, the BICY project:

- developed integrated urban-regional bike planning as strategic part of urban-territorial planning;
- promoted cycling as a key component of public transport;
- rose awareness of local authorities and citizens about bicycle as a good replacement for a car for shorter trips;
- promoted bicycle as a safe and ecological means of transport;
- strengthened the frontrunners experience and develop new and innovative technologies;
- provided quantitative evidence that shift towards cycling improves quality of life;

- promoted a trans-national cross-fertilization between similar projects and partnerships working in the same field

3. Structure and Flow of BICY Project

Here the whole structure of BICY will be described paying attention on the main events and outputs concerning the organization and results of the project. All kinds of documents and reports done during the project will be understood as outputs. All meetings, workshops, conferences, trainings, campaigns, etc. provided during the flow of the BICY will be understood as events.

According to work plan of the project, there are 5 main work packages including all activities starting from preparation of the future project and finishing with realization of actions concept:

- Work package 0: Project preparation
- Work package 1: Project management and coordination
- Work package 2: Communication, knowledge management and dissemination
- Work package 3: Trans-national harmonization of experiences in cycling
- Work package 4: Implementation actions & mutual learning

Each of these parts will be shortly described further.

Two figures – Figures 1.4 and 1.5 – will help to look through the whole BICY project in common. They present graphs which show the progress of BICY in time. The first image is the progress of the project in the perspective of documents' output. The second scheme is the project progress in the perspective of the course of events which were taken place.

X-axis is the flow of time: the duration of the project is 36 months. Y-axis is work packages starting from first of them. Each point on the graphs corresponds to one event or to one output document for concrete action of one work package. The time axis shows, normally, the month when the event took place or the deadline until which document must be ready or activity must be finished.

Special mention must be made that work packages have different numbers of actions (minimum is 2 and maximum is 6). That is why both graphs have the common legend which is relevant for every work package separately. For instance, action 2 according the legend is yellow circle. It means that such kind of circle for each WP is action 2.

The both graphs do not take into account project preparation phase, because this time isn't counted as a part of the whole BICY project duration.

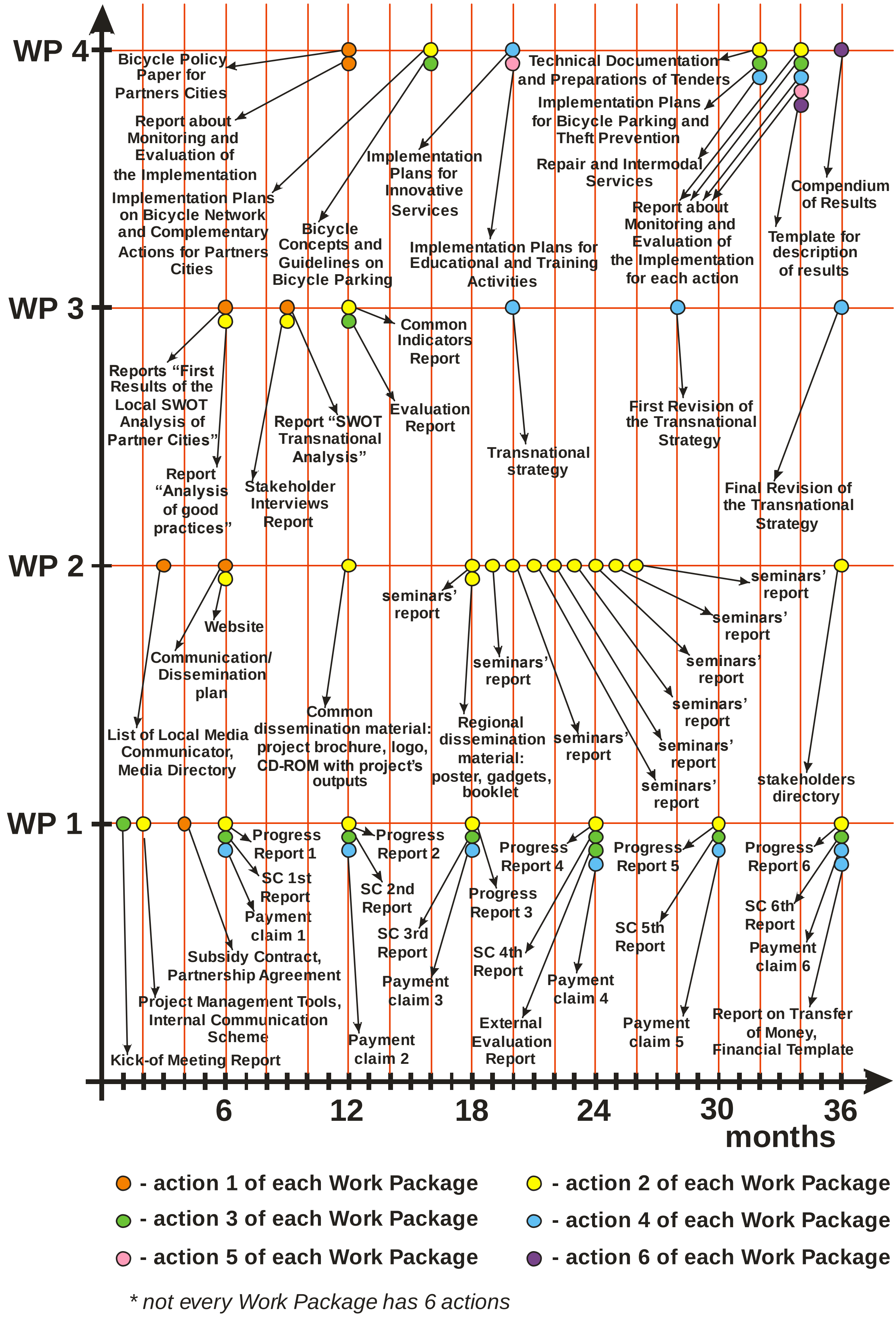


Figure 1.4 – Progress of the BICY project in the perspective of documents output

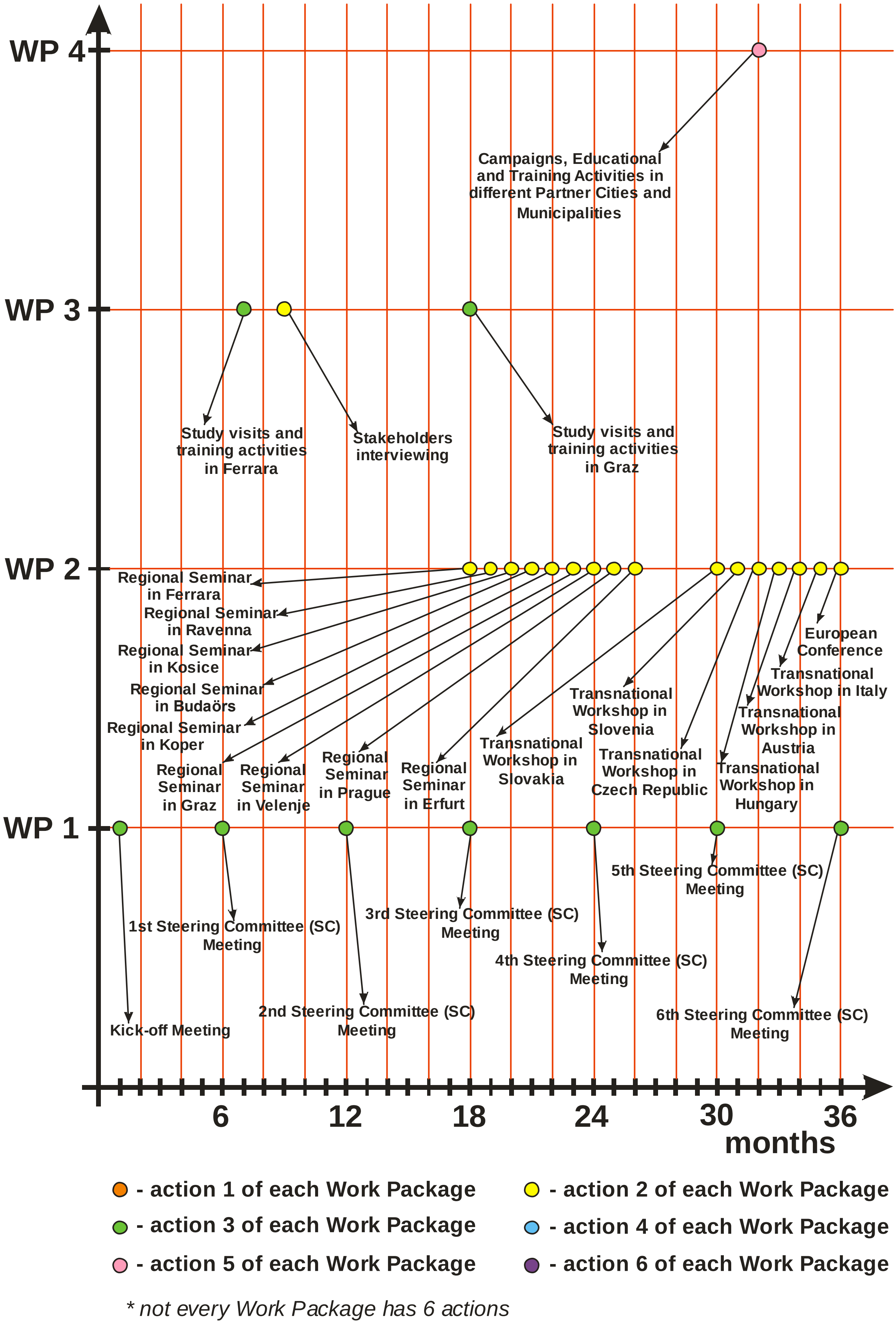


Figure 1.5 – Progress of the BICY project in the perspective of the course of events taken place

3.1. WP0 – Project preparation

As it was mentioned above, the first stage was the project preparation phase. It started at the beginning of the year 2009 (around one year before official BICY project start). 11 partners from 10 regions of 7 different Central Europe countries were involved in the cooperative work. The main aims were:

- to spread the first draft of project proposal and a questionnaire among potential partners;
- to collect all possible feedbacks about this drafts from the future project members;
- to create an interactive environment for potential partners;
- to develop the final version of the project construction;
- to prepare the financial tables.

Also trans-national preparatory seminar took place on the 20th of February of the same year in Ferrara.

3.2. WP1 – Project management and coordination

As the work package name can indicate, the main goal of this phase was to make management and coordination of BICY project more efficient and clear for every partner. According to conditions of CE programme, this stage must be divided into 4 actions which are the following:

1. *Fulfillment of start-up requirements*:

   During the first 4 months from the project beginning two main documents must be signed: Subsidy Contract (between the LP and the Managing Authority of CE programme) and Partnership Agreement (prepared and signed by all partners for establishment of management and coordination rules).

2. *Day-to-day project management, coordination and internal communication*:

   a. The LP selected the Project Manager and the Financial Manager. Each partner in its side designated a Local Project Manager and a Local Financial Manager.
   b. During the first 2 months the LP prepared Project Management Tools package which must ensure the effective project management and the day-to-day coordination.
   c. Also during the first 2 months a special scheme for internal communication was set up. It included new and traditional communication tools. The project website was created with the option to be used by every partner, as well as, to share different kinds of documentation.
   d. Every partner made a number of Progress Reports which were delivered in the end of every 6 months.

3. *Steering and monitoring of the project implementation*:

   The Steering Committee was established by the start of the BICY project. This authority consisted of all project top-management and also of two local managers. Its aims, besides the management work, were also control and monitoring how project activities and financial flows were going on during the whole period of work. To make the work of Committee more efficient, a number of meetings was organized including 1 kick off-meeting in the first project month and also 5 trans-national SC meetings in the end of every half-year during the whole project period. External evaluations of the project implementation were also foreseen.
4. *Financial management, certification of expenditure*:

   As it was mentioned above, the main Financial Manager and a number of Local Financial Managers were selected in the beginning. This group worked together keeping track of all financial questions of project: they prepared certifications of expenditures, controlled that this documents were conformed to the reporting obligations and that PPs received the reimbursements in time, made the financial flow monitoring.

3.3. WP2 – Communication, knowledge management and dissemination

The main goal of this work package was to promote the project output and results. Based on this idea, a detailed communication/dissemination plan was designed at the kick-off meeting and was divided into 2 main actions:

1. *Media communication/ dissemination*:

   Every partner created a list of local media communicators, established relations with them, set up a media directory. The objective was to involve as much as possible national and regional mass media vehicles to cover the activities of BICY project.
2. *Non-media communication/ dissemination and website*:
   a. A stake-holders directory was being created during the whole 3 year of BICY project and included all parties concerned.
   b. Non-media dissemination plan was designed.
   c. A number of public trans-national workshops and also regional seminars were held during the second half of the project period.
   d. European Conference on cycling mobility in Prague was organized in January 2013. This event gathered all the potential stake-holders and other people and organizations working on similar problems.

e. Common dissemination material were produced (number of brochures, CD-ROMs, etc.)
f. Already existing networks as BYPAD and others and also different conferences were used to disseminate information about BICY project and its outputs.
g. The BICY project webpage played an important role. It was not only a place of partners internal communication concerning documentation up- and downloading (as it was mentioned above), but also an open source for public visitors and a platform where actual information and all kinds of materials can be exchanged.
h. A large amount of associated partners (cities, clubs, federations of bikes, etc.) helped to spread information about BICY activities in different ways.

3.4. WP3 – Trans-national harmonization of experiences in cycling

The cycling mobility rate of the total urban trips is one of the important characteristics which helps to estimate a level of bike-usage for a certain place. It fluctuates strongly depending on city or region, and sometimes difference between parameter values for two neighbor regions or countries can be very large. That is why the work package 3 of the BICY had an aim:

- to contribute to harmonization of the cycling mobility rate for the cities and regions which were involved in the project and
- to achievement of the best parameter standards.

To reach the above-mentioned goal, the working groups were created which included representatives of PPs and external experts. These groups worked together to perform 4 actions:

1. *SWOT analysis*:

   Every partner did a SWOT analysis on cycling for its home location. This work was finished until the end of the first half-year from BICY project start. This analysis helped to understand existing situations in the different cities and regions and their advantages and disadvantages, to find the main possibilities for cycling development and also problems which can appear by this development. As a result, the transnational SWOT analysis was done by the DICAM team in the end of the first 9 months.
2. *Common Indicators*:

   The main objective of this action was to determine common performance indicators, including costs-benefits analysis. These parameters helped to find the keys how the cycling mobility and the bicycle traffic can be improved on the different levels: from small cities till the whole CENTRAL EUROPE area. As a consequence, common indicators development was a very important part of the BICY which had influence on the all project realization and, especially, on trans-national strategy.

To establish the common indicators (and also to make costs-benefits analysis included in the final report), a number of activities were done and carried out: analysis of good practices, data collection including mobility survey, OSM data analysis. The first results were available after 12 months from project start, but this work did not stop during the whole project life. Also these results were used during the all events held during the BICY: workshops, seminars, conferences, etc.

3. *Study visits & training seminars*:

   Two trans-national study visits & training seminars were organized in Ferrara and Graz (on the 7th and 18th months of the project) which were visited by representatives of all partners involving in the last stage of the BICY – Work package 4. The aim was to give necessary information about bicycle as a mode of transport and everything connected with it: from basics of policy formulation till monitoring and control of influences after realization of actions concept. All this knowledge was used by PPs during the WP4. Also the trans-national dimension of customer satisfactions was evaluated during the seminars and, according to this evaluation, the common report was elaborated.

4. *Trans-national strategy*:

   All above-mentioned activities were the basis for formulation and construction of a trans-national strategy. This strategy was described in the final paper and explained how to achieve the main aim of WP3 declared above using suitable benchmarking activities and designing required solutions in realistic perspective.

   The process of trans-national strategy development was in progress during the whole project life. The first complete version was prepared until the end of 20th month of the project duration. Then there were also two revisions: according the first feedbacks coming during project implementation described in the WP4 (on the 28th month of project duration) and according the final results in the end of the project.

3.5. WP4 – Implementation actions and mutual learning

In concordance with trans-national strategy getting from WP3, a number of activities and actions were developed as parts of a multilevel implementation approach. Every certain action was conducted in different places in close collaboration with partners. The main goal was to achieve mutually concrete results regardless of special conditions and specifics of a city or region.

The actions were the following:

1. *Policy formulation and strategy building*:

   The first step for all partners was to prepare a cycling policy papers which established the main goals and approaches to achieve them. Some cities and municipalities had already had their policies: they needed only to be revised and adapted. Other ones did this work from the beginning on the basis of existing policies of other partners and of output coming from WP3 (SWOT analysis, common indicators, etc.). This action was finished until the end of the first project year.

2. *Bicycle network and complementary actions*:

   Bicycle networks and their development were one of the important aspects by the cycling policy formulation. Until the end of 16th month from the project beginning, a number of partner cities developed to each its own implementation plan on bicycle network and complementary actions. Near to the BICY project end (32th month), some partner towns had already prepared planning and technical documentations for building new cycle ways and for new signposting about bicycle routes. Also they were ready to implement these construction activities.

3. *Bicycle parking and theft prevention*:

   Equipments of bicycle parking are playing an important role for increasing of bike share in the modal spilt. Theft prevention must be also taken into account: person will never use bicycle as a mode of transport if its two-wheeled vehicle is stolen more than 3 times.

   One of the action ideas was to create a concept (until the end of 16th month) and implementation plan (until the end of 32th month) for bicycle parking and theft prevention for some partner cities.

4. *Innovative services (intermodal and rental; new technologies)*:

   Today an amount of different innovative and technological services for bicycle and cyclists exists which can make a bike usage more attractive. The following examples can be attributed to these services: rental and repair stations, intermodal services (to mix cycling with different means of public transport), exchange parking, bicycles equipped with GPS navigators, etc.

   Until the 20th month of the BICY project, some partner cities and municipalities prepared their own concepts of different innovative bike services which can be installed there. Then (until the 32th month), on basis of these concepts, a number of implementation plans for proposed services was appeared.

5. *Raising of awareness, campaigns and education, training and target group activities*:

The main goal of this action was to raise the image of cyclists and cycling. To reach this aim a number of events was held by PPs.

By analogy with the previous point, concepts of activities were created by partners until the 20th month of the project. And until the 32th month, all scheduled activities took place in the partner cities and municipalities. There were, for example, educational meeting in Graz and Budaörs, promotional campaigns in Prague, Kosice and Erfurt, etc.

6. *Compendium of implementation action results*:

   In common, the compendium is an amount of materials concerning activities and their results according to implementation actions of the WP4. It includes articles, CD-ROMs, files which can be downloaded from BICY website, etc. They were prepared, published and collected during the whole project duration. A structure of the compendium was made according the above-described actions of WP4: each action corresponds to each chapter. To make a system and to get an in the right way described information about undertaken activities, a common template for results description was developed and given to each partner.

   The compendium had both local and trans-national importance. On the one hand, for example, with help of information uploaded on the website, people were informed about activities and events which were planed and accomplished by PPs, also could find results reports. On the other hand, the trans-national compendium was presented on the Final European Conference in the end of the BICY project. This amount of examples how to promote cycling mobility and to improve bicycle culture in different regions of CE can be a good handbook for stakeholders and decision-makers in the whole European Union by making cities and regions more liveable and accessible.

Regular monitoring and evaluation of the implementation were carried out for each WP4 action and, consequently, appropriate reports were written until the end of each action. It means that the results reached by certain city or region were analyzed according to a common trans-national indicators grid.

4. Work Package 3 as application field of theses

The common description of work package 3 was given above. But as the main focus of these doctoral theses is directly connected with WP3 and, especially, with some of its activities (*Analysis of good practices* and *Common Indicators*), this part of BICY project will be considered more detailed in this subchapter.

4.1 Common relationship between WP3 activities

As it was mentioned above, the transnational strategy is the core output of the whole work package. This paper is a basis to plan and realize an amount of implementation actions which must ensure harmonization of cycling level of the partner cities and regions and at the same time achievement of necessary standards.

A number of activities according to the project work plan was done to develop this strategy. They included a compared analysis of existing and new bicycle policy papers, SWOT analysis, analysis of good practices, interviews with stakeholders and common performance indicators (or also called Common Indicators). The influence of all these components on the transnational strategy development is shown in the Figure 1.6 with black arrows. The tasks of the work package 3 have green background on the picture, of work package 4 – yellow (transnational strategy is also a part of WP3, but it is marked with orange color as a main aim gathering all influences).

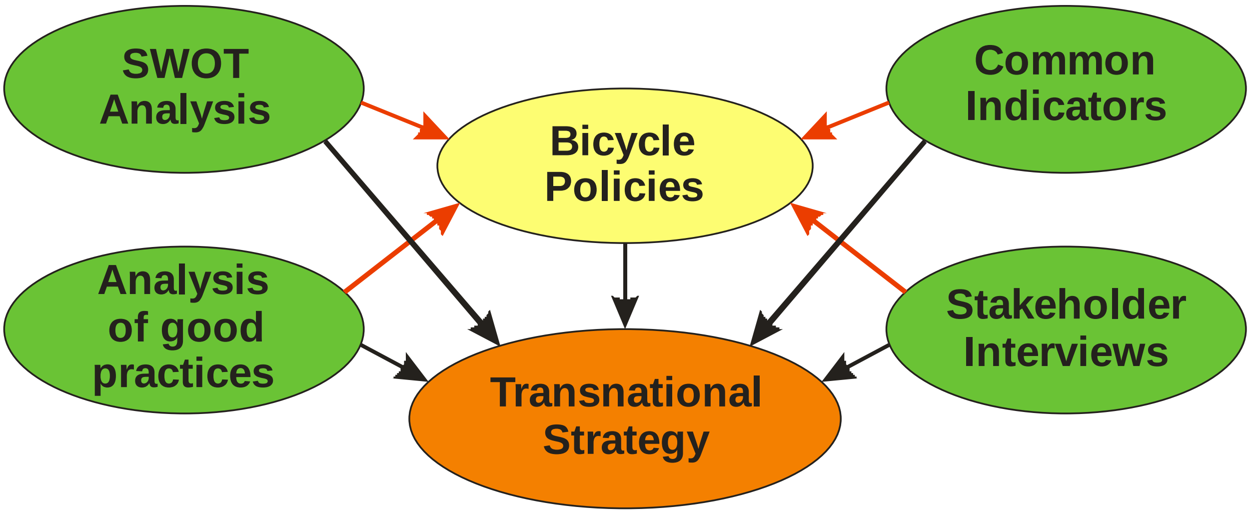


Figure 1.6 – Influences of the planned activities on the transnational strategy development

All above-mentioned activities have not only impacts on the strategy, but also relations between each other and impact on each other. For example, in the Figure 1.6 it is possible to see red arrows which show what kind of direct connections bicycle policies of partners have with others: they were based largely on the tasks of WP3. The relationships of the WP3 activities will be described further.

The aim of *the SWOT analysis* is to assess the situation with cycling mobility in each partner region, to find out its advantages and disadvantages, also potential problems the future development of cycling could face. The SWOT transnational analysis is a final result of a common cycling state in the perspective of all partners. Between all WP3 activities it had a direct linkage with good practices analysis: on the one hand, it was carried out looking through the prism of SWOT analysis results, on the other hand, it helped partners to do SWOT analysis by giving a number of main directions and focuses. SWOT analysis had also a direct influence on Common Indicators: its information about existing situation was an orienting point for organization and carrying out data collection activities which were one of the necessary conditions for indicators assessment.

The analysis of good practices (also called *State of the Art*) was made with the following goals:

- to search and analyze of the results from different similar projects which are in progress or were realized within the European Union;
- to look through research methodologies of cycling mobility and its indicators for finding possibilities to use or to adapt them for BICY project needs;
- to find leader cycling countries and cities (with the high bicycle usage), to analyze their network systems and mobility situations comparing each other and with partner cities, to establish their main tendencies in the cycling development

As it was written above, analysis of good practices is connected with the SWOT analysis. It also has a direct linkage with Common Indicators: realization of all its goals has influenced the development and calculation of cycling indicators.

The interviews with stakeholders were carried out to get a feedback and different new ideas and opinions related to the BICY project from the experts and cycling stakeholders. This activity had a direct influence on common performance indicators: it helped to correct the approaches used for measurements, calculations and analysis.

The main objectives of *the common performance indicators (or Common Indicators)* were to collect data about cycling mobility and its potential, to make analysis and assessment of this data, to search patterns and to develop models based on obtained information, and, finally, to make prediction for the future of bicycle usage and other relevant parameters. Its relationships with other relevant WP3 tasks have already been described above. Special mention must be made that Common Indicators had the strongest linkage with analysis of good practices.

Of course, there is a number of indirect relations between all above-mentioned WP3 activities. Figure 1.7 presents only direct interactions between them.

Common Indicators have a more complex structure as compared with SWOT analysis, analysis of good practices and interviews with stakeholder. That’s why it will be described separately in details further.

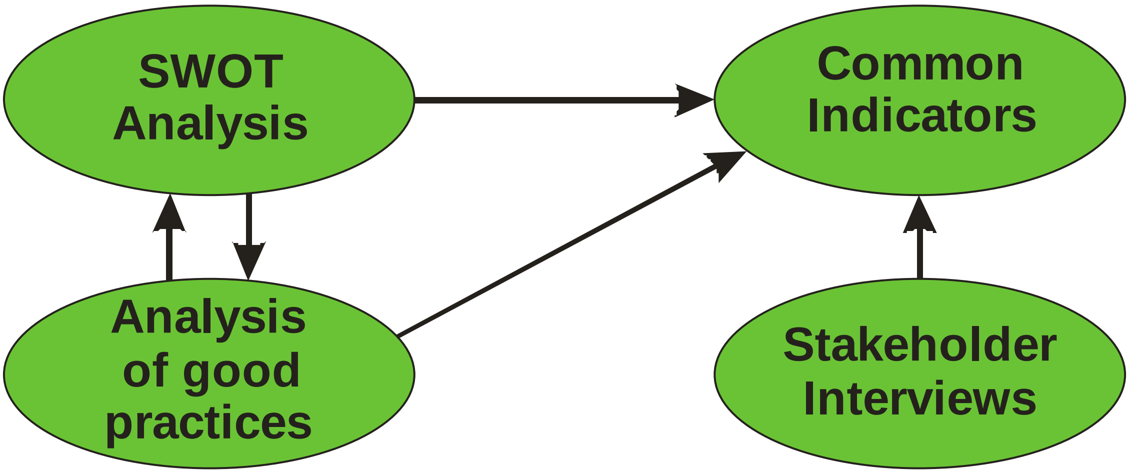


Figure 1.7 – Direct interactions between WP3 activities

4.2 Structure of Common Indicators identification

The main aim of Common Indicators and its relations with other WP3 activities were described above under the point 4.1 of this sub-chapter. The issue here is its structure.

A flow-chart of current WP3 task is presented in the Figure 1.8. Different colors of boxes in the block scheme mark different stages of Common Indicators. In total, there are three steps of indicators activity. Block *transnational strategy* in the Figure 1.8 presents a separate activity and the main result of the whole WP3 action which is based to a great extent on the Common Indicators and its results.

Green boxes are the first stage of the considered WP3 activity. It includes three primary components: *data-collection*, *mobility survey* and *infrastructure data collection*. All these elements had the common goal to collect necessary information for further steps of Common Indicators.

Blue Boxes in the Figure 1.8 represent the second stage of Common Indicators – the first level of data analyses and calculations of the results. The common objective of this phase was to get a number of outputs which will be relevant not only for the next step of current WP3 activity, but also for definition of transnational strategy. Elements of this stage are *survey evaluation*, *infrastructure analysis* and *indicators calculation*.

Yellow squares of the flow-chart identify the third and final stage of Common Indicators activity – the second level of analyses and evaluations. The basis for this phase was data and results got on the previous stages. The main components are *potential demand assessment*, *indicators analysis* and *cost-benefit analysis*. The aim of this step was to prepare a number of outcomes which would have a great importance for the trans-national strategy formulation.

Every component of each stage will be considered further in details.

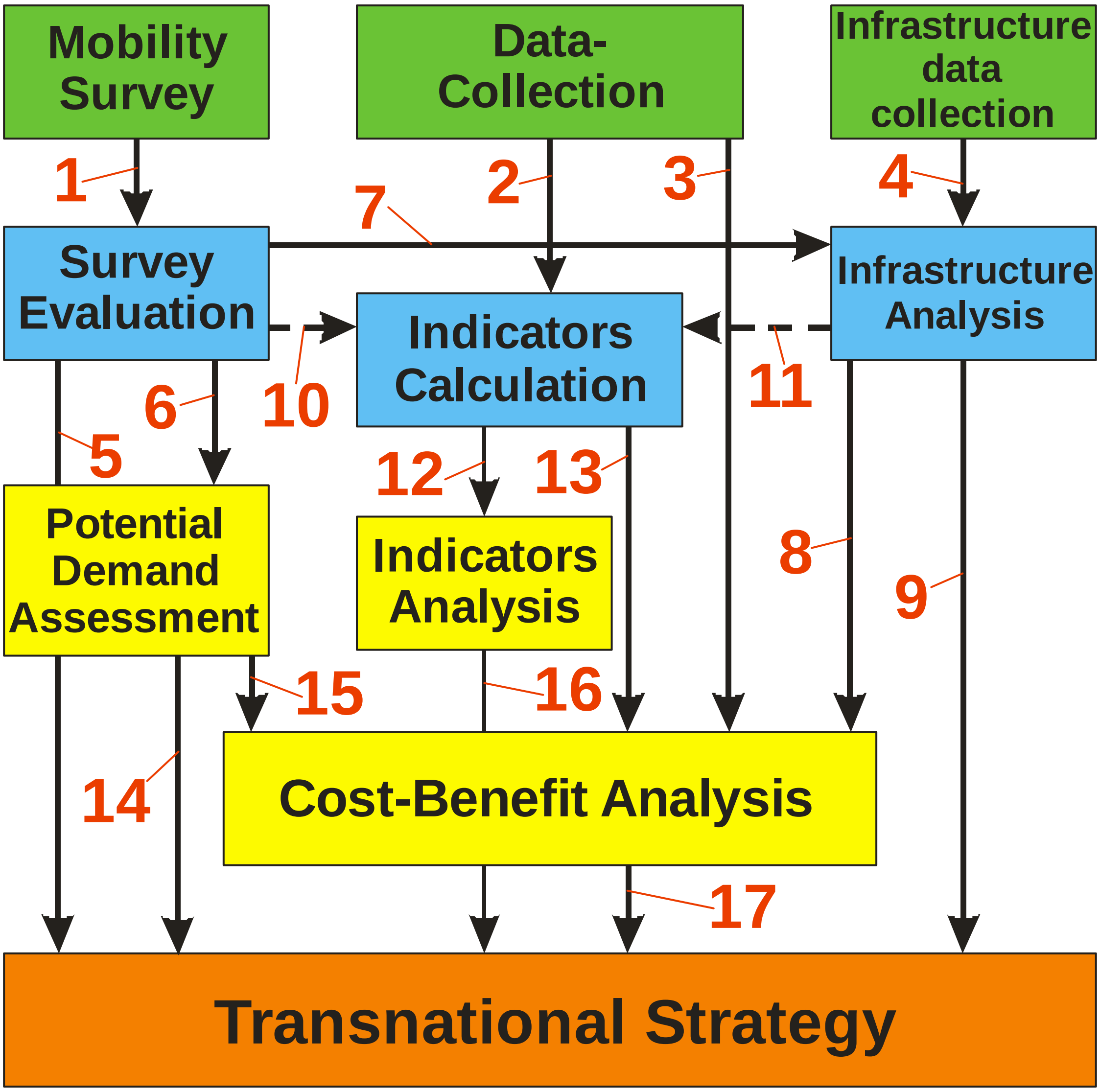


Figure 1.8 – The basic flow-chart relationship of Common Indicators

4.3 Stage 1 of Common Indicators activity

As it was mentioned above, the first stage has 3 main elements which contributed to the collection of different data. They are *data-collection*, *mobility survey* and *infrastructure data collection*.

Data-collection

The data-collection is an amount of common quantitative data about infrastructure, population and external costs made by each of the partners. The special data collection form was created and sent to every PP. It countained a list of the following data:

- reference area (km^2);
- population, also its sex and age distribution;

- population density (1/km^2);
- road length (km);
- exclusive bicycle path length (km);
- length of road with traffic calming (km);
- length of pedestrian areas where bikes are allowed (km);
- number of bike parking or racks;
- total number of light-rail/ local-rail/ metro stations with and without bicycle parking;
- total number of severe road injurious and road fatalities caused by cars per year;
- "construction" costs of segregated bike path by markings (€/km per lane per direction);
- construction costs of new segregated bike path which are physically separated (€/km per lane per direction);
- costs for installing 10 bicycle racks (€);
- various data of local public transport operator;
- modal split;
- modal split by km (km driven by each mode per day per person);
- transport-related energy consumption or CO_2 emissions per person per day or per year.

Also some additional useful data concerning partner cities and regions was gathered by PPs themselves and UNIBO-DICAM team. For example, there was an information about topography, weather conditions, etc.

As it could be seen, the information gathered was various enough, and it required a lot of efforts from the partners to search it using different sources. Unfortunately, this step of data collection didn't go very smoothly. There were several problems for a number of reasons. They were described in details in the annex A of Common Indicators Report [13]. In general, the common issues were the following:

- a number of data is unavailable or incomplete by different reasons;
- some types of data obtained from one partners are incompatible with the same types from another partners which could not provide possibility to compare them;
- some data from BICY project partners were collected using non-representative methods and, as a result, could not be used.

Mobility survey

The mobility survey was carried out by every partner. This instrument provided data about the travel behavior of the inhabitants. Data also included opinions about the present transport supply and readiness of citizens to change their usual mode of transport to another more sustainable one. A detailed description of the survey preparation and carrying out will be done in the chapter 3.

Infrastructure data collection

The infrastructure data collection was a gathering of GIS data based on OpenStreetMap (OSM). Thus, comparable information about transport network of all partners' cities and regions including bicycle equipment and facilities was obtained. The download process of GIS data from its source will be described in the chapter 4.

4.4 Stage 2 of Common Indicators activity

The components of the second stage are *survey evaluation*, *infrastructure analysis* and *indicators calculation*. They returned the first results of data processing.

Survey evaluation

As it is shown on the flow-chat in the Figure 1.8 with an arrow 1, *survey evaluation* is based on the mobility survey which provided a huge amount of data. Processing of this data allowed to assess a number of parameters including modal split and also to calculate effects of the possible implementation of various cycling improvement scenarios. The major part of survey evaluation results had a direct influence on transnational strategy (arrow 5, Figure 1.8). But also there were some outputs (for example, scenario demand) which connected with one component of the third stage called potential demand assessment (arrow 6, Figure 1.8). More deep survey evaluation will be explained in the chapter 3.

Infrastructure analysis

Infrastructure analysis was carried out directly after GIS data collection (arrow 4, Figure 1.8). This analysis included building of classification for the necessary elements of transport network, creation of different filters for data assessment, construction of output data and plots, etc. Other component of the second stage (survey evaluation) had an influence on the process of infrastructure analysis (arrow 7, Figure 1.8): gathering of information about scenario requirements is important to find out all possible solutions for infrastructure improvement. Analysis' results helped directly to the transnational strategy development (arrow 9, Figure 1.8). Also one part of outcome concerning scenario infrastructure modification was a necessary precondition for the cost-benefit analysis on the final stage (arrow 8, Figure 1.8). Infrastructure analysis will be considered in details further in the chapter 4.

Indicators calculation

Indicators calculation is one of the key elements of Common Indicators, because it returned a number of parameters which showed the full picture about the current cycling mobility situation in the certain partner city or region and, thereby, made it possible to compare different cities. It is a background for the indicators analysis (arrow 12, Figure 1.8). Also this component of Common Indicators has an influence on cost-benefit analysis (arrow 13, Figure 1.8).

According to the BICY project, all cycling indicators can be divided into three groups: *standard indicators*, *bicycle use indicators* and *BICY indicator*.

The assessment of bicycle infrastructure is made with the help of *standard indicators*:

- *Cycling index* = $\frac{km\ of\ cycle\ tracks}{number\ of\ citizens}$
- *Network coverage index* = $\frac{km\ of\ roads}{km\ of\ cycle\ tracks}$
- *Network density index* = $\frac{km\ of\ cycle\ tracks}{area\ in\ km^2}$

BICY index was created by UNIBO-DICAM team with the aim to show better the attractiveness of a bicycle infrastructure for road users which has a direct influence on cycling usage. In the same time this index is simple to be calculated. It consists of two parts:

- *Coverage index* = $\frac{km\ effective\ bike\ path}{km\ road\ network}$
- *End-point index* = $\frac{number\ of\ public\ bicycle\ parkings}{number\ of\ public\ car\ parks}$

It is logical that to make a trip from one point to another a cyclist needs to have a bike path and bicycle parking. That’s why conjunction “and” in previous sentence can be interpreted as multiplication:

- *BICY index = Coverage index x Endpoint index*

If BICY index equals zero – the cycling infrastructure of a city or a region is underdeveloped.

If BICY index equals one – a city or a region has achieved an ideal state of the cycling infrastructure.

Bicycle use indicators are parameters which show directly cycling mobility rate in comparison with above-described “infrastructure” indicators. The most well-known characteristic from this indicators group is the modal split calculated for the concrete trip length (for example, less than 5 km) and for the certain trip purpose (for example, from home to work):

- *bike share* = $\frac{number\ of\ trips\ by\ bike}{total\ number\ of\ trips}$

But, unfortunately, bicycle share has specific flaws which must be avoid. Additional indicators help to do it and, as a result, to get a fuller picture about bicycle usage:

- $km\ bike\ share = \dfrac{km\ by\ bike}{total\ \ km\ driven}$

- $car\ km =\ km\ driven\ by\ car\ \ per\ citizen\ \ per\ day$ (for trips below 5 km)

Indicators calculation was based on the data got from each partner (arrow 2, Figure 1.8). As the information received from BICY partners had a number of problems mentioned above, two other components of the second stage – survey evaluation and infrastructure analysis – and their outcomes gave an opportunity to correct and to calibrate data or, sometimes, to fill gaps in them. In the Figure 1.8 these links were shown with arrows 10 and 11.

All presented indexes will be explained more thoroughly and discussed below in chapters 3, 4 and 5.

4.5 Stage 3 of Common Indicators activity

The final stage includes three elements: *potential demand assessment*, *indicators analysis* and *cost-benefit analysis*. They provided the final results of Common Indicators identification – an essential background for transnational strategy.

Potential demand assessment

Potential demand assessment based on the results got from survey evaluation (arrow 6, Figure 1.8). It is also possible to identify this component as a part of the survey evaluation. Potential demand was calculated and evaluated with the help of scenarios data. It showed a possible increase of cycling and public transport usee by implementation of a number and a variation of different measures. During this activity the characteristics of a “car lover” and a “potential bike user” were also estimated and the shares of such road users for every partner were calculated.

Potential demand assessment is connected with the cost-benefit analysis (arrow 15, Figure 1.8). And it has also a direct influence on the transnational strategy development (arrow 14, Figure 1.8).

Indicators analysis

Indicators analysis is an analysis based on a number of indicators calculated on the previous stage (arrow 12, Figure 1.8) and described above. The main goal was to search any correlations between calculated indices and bicycle mode share and, if such ones exist, to analyze them for:

- finding an explanation of influence made by different factors and
- development of models which can make reliable prediction of cycling levels for the future

Indicators analysis had an effect on the transnational strategy formulation (arrow 16, Figure 1.8). Its results were described in details in the Common Indicators Report [13] and, especially, in the annexes I, J and K. All parts of theses will more or less return to them, but the main discussion will be done in the last chapter.

Cost-benefit analysis

Cost-benefit analysis is a calculation and estimation of the costs and benefits which will be obtained by potential increase of cycling rate. This activity is vitally necessary for the transnational strategy formulation (arrow 17, Figure 1.8). The common idea was to estimate profits which could be obtained by the reduction of accidents, energy-consumption and emissions due to growth of cycling mobility. For this reasons an amount of information was gathered within the data-collection action (arrow 3, Figure 1.8) including:

- external costs: road injuries and fatalities caused by car drivers per year, transport-related energy consumption or CO_2 emissions per person per day or per year, urban density;
- cycling infrastructure costs: cost of 1km of cycle track (physically segregated and segregated by markers), costs for installing of 10 bicycle racks.

As it was stated above, the information about necessary infrastructure modification for the cost-benefit analysis was received from the infrastructure analysis (arrow 8, Figure 1.8). Indicators calculation gave data on the situation with cycling mobility, especially, concerning actual bike share (arrow 13, Figure 1.8). Potential demand assessment concluded the information on future changes in bicycle usage for different scenarios (arrow 15, Figure 1.8).

Conservative results of the Common Indicators component are the following:

- cost-benefit ratios of up to 12:1 for eastern partners, saving lives and millions of Euros per year, by investing in bikeways;
- substantial carbon reductions.

To find more detailed information about cost-benefit analysis, methods used for calculation and estimation, full list of results look at the Common Indicators Report [13].

5. Main objectives of theses in the perspective of BICY project

As it is seen from theses title, the main focus of the whole doctoral dissertation is to create and to develop methods of automated analysis which evaluate cycling infrastructure and demand for bicycle. This process will be divided into two main elements:

1. creation and development of special method for bicycle demand assessment and

2. creation and development of special method for bicycle infrastructure assessment.

The main basis for the first method was *mobility survey*, *survey evaluation* and *potential demand assessment*. All these three components of Common Indicators and everything concerning them were presented above. These tasks were used as realization platform for creation and testing an automated analysis method which serves to make bicycle demand assessment. The following steps of its development were carried out:

- survey evaluation and potential demand assessment: their necessity and their requirements;
- search of existing survey methods to establish their similarities and differences, their advantages and disadvantages;
- mobility survey carried out during the BICY project;
- design and development of program which makes processing of questionnaires more automated and allows to create a database of obtained information;
- handling of questionnaires data with the help of automated analysis approach which is a part of above-mentioned package and is oriented to required output results.

The method for bicycle demand assessment, its bases, its design and development will be described in the chapter 3.

The development of the method for bicycle infrastructure assessment was based essentially on three following components of Common Indicators: *infrastructure data collection*, *infrastructure analysis* and *indicators calculation*. The relationships between these three elements, their role and importance for the BICY project were described and explained above in the actual chapter. Also, by analogy with the first method, these tasks were a platform on which ideas for development of this automated analysis method can be realized:

- what kinds of infrastructure data are required to make an assessment in the perspective of cycling mobility;
- where and how this infrastructure data can be gathered;
- how obtained data can be processed and classified;
- how the whole information can be analyzed and outputs can be presented;
- what kinds of opportunities exist for the future implementation and development of this method in other projects and researches.

The method for bicycle infrastructure assessment, its development and other important questions concerning it will be considered in details in the chapter 4 of theses.

Also the analysis of good practices will be considered in the separate chapter of this doctoral dissertation. The main goals of this WP3 activity were described above. But here it will be presented as a previous step before starting development of two automated analysis methods. It is needed mainly not only for understanding an actual situation with cycling mobility and finding leaders between countries and cities, but also for analysis of the best existing projects connecting with bicycle traffic and infrastructure and of their assessment methods and approaches. All these actions will help to determine research directions for development of above-mentioned methods and also to find reference points which must be known to start creation of these automated analysis approaches.

Chapter 2 – Analysis of Good Practices

1. Historical development

Chapter 2 begins with a historical overview of cycling development. This subchapter shortly describes a history of bicycle traffic and infrastructure starting from appearance of the first cyclists until the end of the XX century.

1.1 The earliest years of cycling development

A bicycle was firstly introduced at the end of the nineteenth century. At that time it wasn't necessary to build special facilities for this mode of transport because the existing infrastructure (main roads, dikes, etc.) could already be used. Besides, its structure was much better in comparison with the today's situation: a route composed normally of the shortest lines from steeple to steeple. Primarily cyclists themselves tried to improve the comfort of cycling. The first attempt in this way was a pneumatic tire created by Dunlop.

The appearance of a car had cardinally changed the aligning of forces on a road. Of course, the number of cars right after their invention was extremely small: for example, on the Dutch roads in 1930 there were 67 000 cars versus 2.5 million bicycles. The main thing that a car brought with it was the change of the road network. Also, the amount of accidents increased due to a increased difference of speeds and masses of the various road users. Thereby, new roads and other road network facilities were designed taking into account the following:

- bicycle traffic should be separated from motorized traffic;
- car flow must not interrupt;
- road network should prevent physical conflicts between bicycles and cars.

The second development step started in the fifties: the first cycle tracks for recreational use were constructed. They were constructed in attractive recreational areas and financially supported by specific grants. There was no goal to connect them with the "utilitarian" cycle tracks. Also this cohesion was not included in the grant system.

The interest towards cycling mobility increased a lot in the seventies because of the energy crisis and environmental awareness rise. Demonstration project was carried out in two Dutch cities: Den Haag and Tilburg. The results presented such important for cyclists aspects as directness, short stops at traffic lights and a level of comfort while cycling. It revealed the significance of good bicycle infrastructure facilities and put forward the issue of road safety. Another demonstration project in Delft brought other interesting outcomes: the competitiveness of a bike as a mode of transport can be increased by the road network improvement. On the other hand, there were situations when cities with high rate of bicycle usage lost the bike share in total modal split by ignoring of further cycling developments. In other words, complex cycling development policy for each municipality or region starts to play an important role.

As a result, since the seventies every country and city had been standing at the choice: either to follow the existing policy or to change their vision towards a bicycle-friendly system. For example, many cities in the Netherlands and also some cities in other European countries have adopted bicycle-friendly policy. However, many other municipalities in Europe did not introduce pro-cycling policies at all or introduced them much later. This would partially explain the large difference in bicycle usage between the Netherlands and others European countries.

1.2 SHT study

Historical study performed by the *Stichting Historie der Techniek* (SHT, Foundation for the History of Technology) came to the similar conclusions. The report with the results of the study was published in the April 1999.

The aim of this historical study was to make a research of cities and regions with different levels of bicycle use rate in the Netherlands and surrounding countries. Between Dutch cities Amsterdam, Eindhoven and Enschede were chosen as cities with high bike share in total modal split. Urban region Kerkrade-Heerlen in South-East Limburg was also taken in account, were the cycling level was relatively low. In theneighboring countries five cities were considered:

- cycling city Copenhagen;
- bicycle-free city of Manchester;
- Antwerp, Basel and Hannover as cities with an average rate of bicycle use.

City records since 1900 of all chosen places were researched and analyzed in the perspective of traffic policy and urban development. Especially, a lot of efforts were done to reconstruct comparable trend lines of bicycle usage in each of these cities (Figure 2.1). These lines present clearly the historically rooted differences in cycling development.

The trend lines in the Figure 2.1 are based on various data sources, assumptions and estimates, which may influence the overall result. The dashed lines indicate that estimates have played a more important role. During the Second World War there were no any feasible estimates.

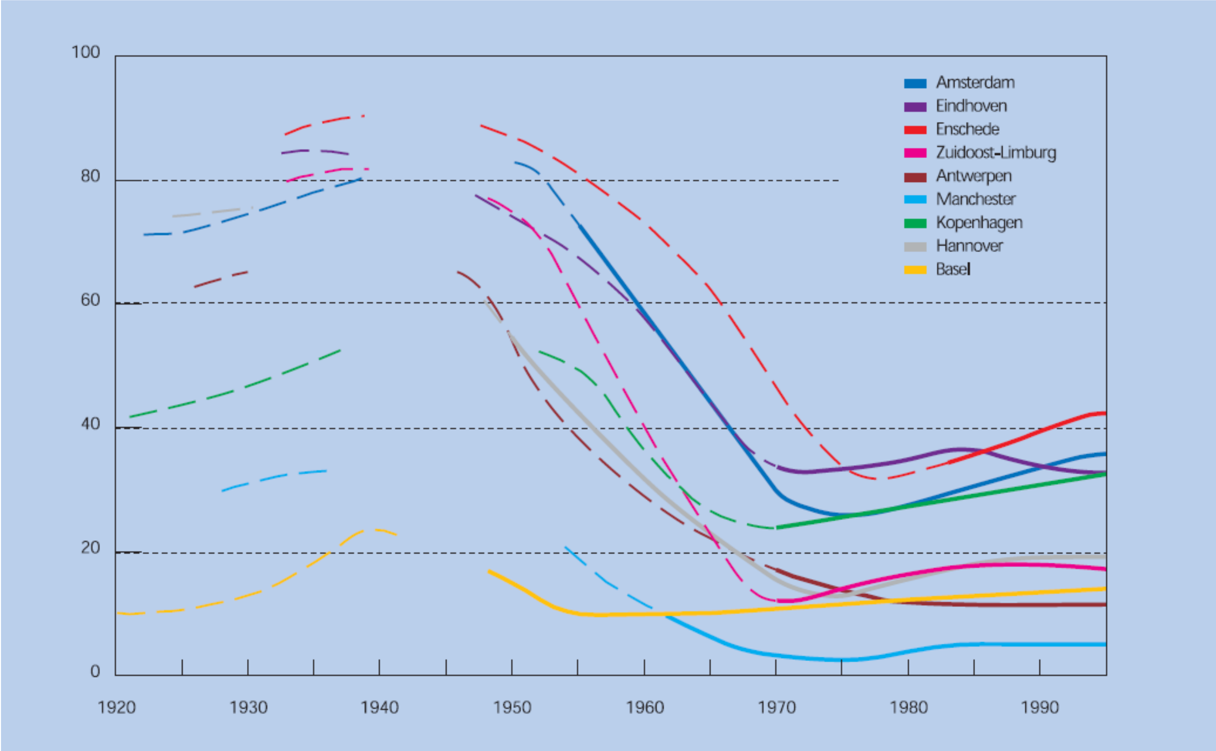


Figure 2.1 – Reconstructed trend lines of bicycle shares in the total number of car, bicycle, moped and public transport journeys, in %, 1920-1995 (source: Fietsberaad Publication № 7)

It is possible to compare all nine trend lines with the help of adapted method. Starting from 1900 bicycle began to be not only leisure product, but also utilitarian product for all classes of society. Until the 1950s all cities had relatively high share of bicycle use: each place on its level. The difference in cycling between cities before the Second World War was caused less by motor car utilization and more by the degree of walking and by early development of public transport (as it was in Manchester). Fast development of car industry in the 1960s created conditions for the rapid decline of cycling level. Only from 1970s a stabilization phase or a renewed development of bicycle usage started.

Looking at the Figure 2.1 it possible to notice the following:

- parallels in the trend lines;
- continuous difference in the cycling level between cities during the whole period;
- difference in the intensity of the growth and fall of the bike share lines.
- Conclusions of SHT-report explain clearly the final differences between the cities in the 1990s: Amsterdam, Eindhoven, Enschede and Copenhagen were leading cities in case of bicycle usage – more than 30 %. In these municipalities bicycle users had never changed

their own mode of transport to another ones (also in the situation of public transport development). Bicycle traffic was always a part of transport policy: "Accepting the cyclist as a "normal" traffic participant with equal rights in the '50s and '60s has been (...) a crucial factor: the realization of a motor car infrastructure is not at the expense of the cyclist; the collective bicycle picture is fairly positive and especially "rational"." [4, page 9].

- South-East Limburg and Hannover had a mean level of cycling – ca 20 %. These two cities had transport and spatial infrastructure development policy which was adapted step-by-step to the growth of the motor vehicle traffic and satisfaction of its needs.
- Antwerp, Manchester and Basel were the cities with a low bicycle share – ca 10 % or below. Their traffic policies were initially oriented on motor vehicles and also on public transport development. That had reduced at the earliest steps the cycling level (especially in Manchester): "The decline which was the result of the arrival of the motor car continues uninterrupted and without "brakes", because all relevant influencing factors are pointing in the same direction: a negative collective picture, a strong car-oriented policy, realization of a large-scale car infrastructure, strong suburbanization (...)." [4, page 9].

The main conclusions made by SHT-researchers were that the differences in cycling levels between all mentioned cities in the 1990s could be explained by two aspects: on the one hand, the results of local infrastructure and traffic policies implementation and, on the other hand, the role and importance of bicycle as a mode of transport in the whole transport system of each city. But this explanation must be also considered in the long-term perspective over the decades. In such a case traffic policy had a significant, continuous influence on the final situation shown in the graph compared to infrastructure policy and bicycle role determination which changed the system by slow degrees. Thereby, decisions made in the 1950s and 1960s have an effect on the present time situation but to various degrees.

The SHT-report conclusions can be also clarified by two extremes. For example, Antwerp, Basel and Manchester pursued the policy aiming at support of motor car usage. These policies represented bicycle as a limited mode of transport. Vice versa Amsterdam, Eindhoven, Enschede and Copenhagen always paid a great attention to the actual traffic situation by developing the future policies. And, thereby, cyclists and their needs were always taken into account.

Finally the most remarkable result of this study was that strong and long-lasting cycling culture was likely to advance bicycle use nowadays. For example during the 50th, Antwerp and South-East Limburg showed higher rates of bicycle use than Copenhagen. Afterwards cycling rate in South-East Limburg fell below 20% while Copenhagen raised its bicycle share to over 30% in the recent years. It appears that higher level of cycle use in the past does a favor to the increase of cycling today, but it is not a precondition. Though, the majority of the examined cities with a low cycling culture remained at a low level of cycle use.

2. Dutch bicycle use in a European perspective

It is undoubtful that the Netherlands is the leading cycling contry in Europe. However when comparing the figures it must be emphasized that there is a lack of reliable international and European statistics which would give compatible data about bicycle usage per country. A large compilation of data relating to the bicycle use in different countries and cities in Europe was made in 2006 generally according to the information available in the Internet. Figure 2.2 presents one of the main indexes concerning cycling – bicycle share in all journeys by inhabitants for various countries.

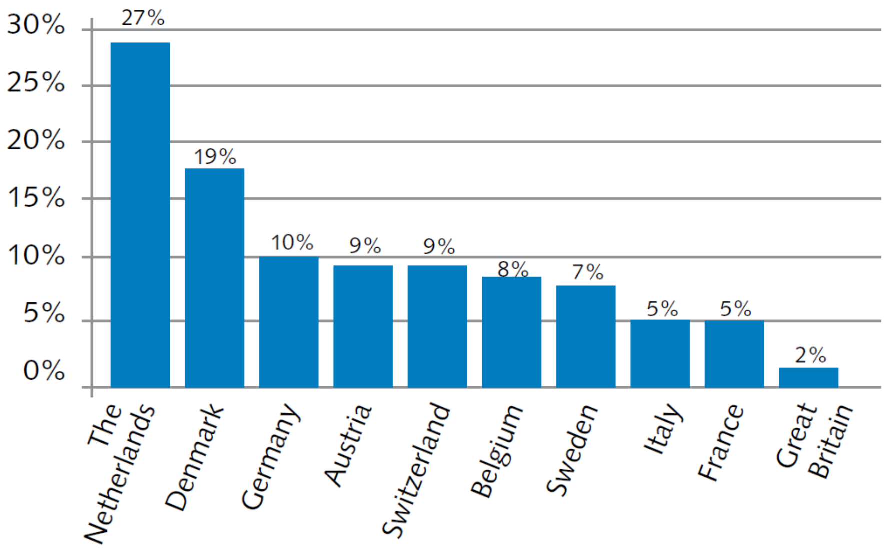


Figure 2.2 – Bicycle share in all journeys in some other European countries (source: Cycling in the Netherlands 2009)

The Netherlands is the onliest country in Europe where the number of bicycles exceeds the number of population (Figure 2.3). As it is shown in the graph, every Dutch resident has in his/her ownership 1.11 bicycles on average. Also this country takes a leading position in bicycles´ selling volumes: for example, 1.2 million bicycles were sold in 2005 considering the fact that Dutch population was only 16 millions. Looking at absolute terms, more bicycles is bought only in the countries with sufficiently higher population: Germany – 4.9 millions for 82 million residents, France and Great Britain with much more comparable number of population (around 60 millions) – 3.2 and 2.5 million bicycles respectively.

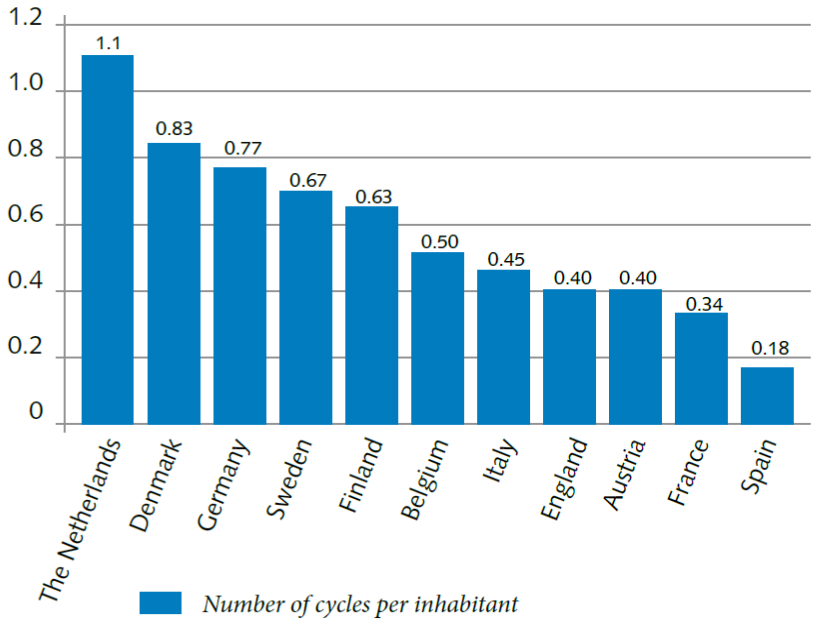


Figure 2.3 – Bicycle ownership in a number of European countries in 2004 (source: Cycling in the Netherlands 2009)

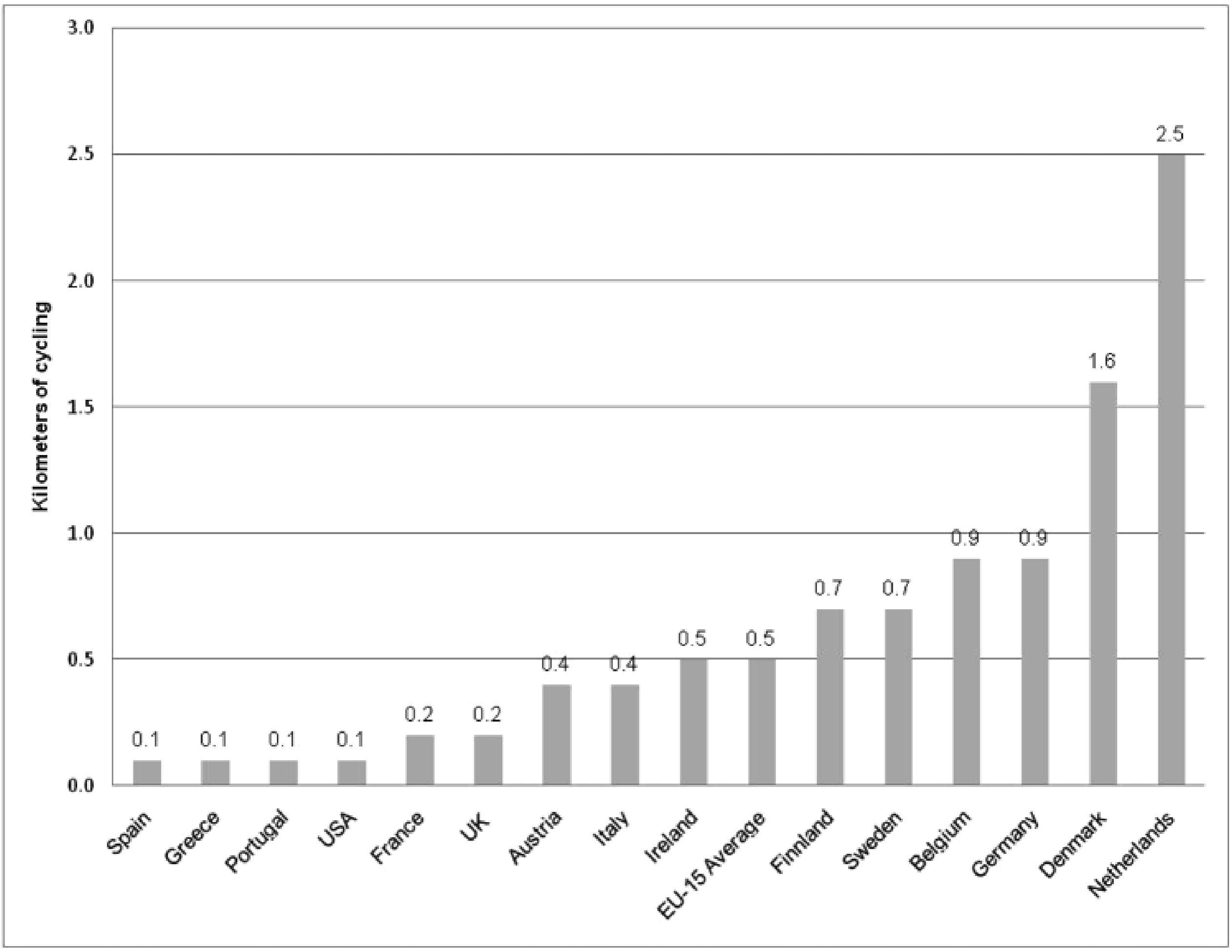


Figure 2.4 – Kilometers cycled per inhabitant per day in Europe and the USA (source: Pucher, John and Buehler, Ralph (2008))

The European Conference of the Ministers of Transport in 2004 made an estimation of kilometers ridden by bicycle per day averaging over the number of inhabitants for each European country. The results are shown in the Figure 2.4. The value of this parameter has relatively great variety from country to country: the smallest value is 0.1 km of cycling per inhabitant per day in Spain, Greece and Portugal and the biggest one is 2.5 km in the Netherlands. Directly after the leader come Denmark and Germany with 1.6 km and 0.9 km respectively. The USA with 0.1 km and the United Kingdom with 0.2 km are on the last positions in the list.

As it was mentioned above, three leading countries according to the average kilometers cycled daily per capita are the Netherlands, Denmark and Germany. For these countries a trend from 1978 until 2005 was calculated with the help of national data analysis (Figure 2.5). As shown in the graph, the considered cycling index grew during the whole period: for the Netherlands – from 1.7 to 2.5, for Denmark – from 1.3 to 1.6, for Germany – from 0.6 to 1.0. Also special mention must be made that the fastest increase of cycling in the Netherlands and Denmark started from the middle of the 1970s until the early 1990s. For the comparison the situation in the United Kingdom was presented, where average kilometers cycled per inhabitant per day fell down during the same period: from 0.3 to 0.2.

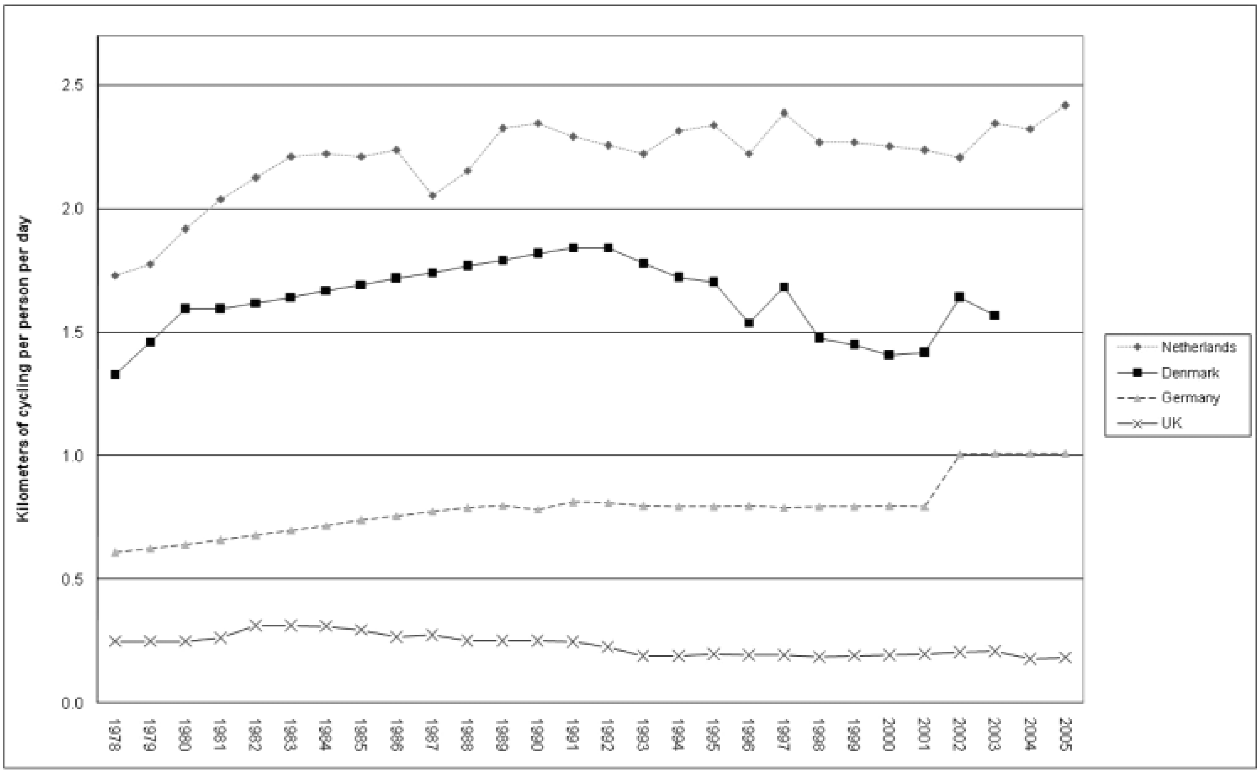


Figure 2.5 – Trend in kilometers cycled per inhabitant per day in the Netherlands, Denmark, Germany and the UK, 1978–2005 (Source: Pucher, John and Buehler, Ralph (2008))

As all the graphs show, the country of the highest bicycle use in the world is the Netherlands. In the Table 2.1 the evolution of figures on passenger kilometers by bicycles and by car in the Netherlands is presented. However, the number of deceased cyclists and car occupants reduced with the increase of the amount of passenger kilometers. That indicates the efficiency of road safety policy having been implemented during these years.

Table 2.1 – Number of passenger kilometers and number of deceased cyclists and car occupants in 1980 and 2001 (source: Cycling in the Netherlands 2009)

	1980	2001	2005
Passenger km by bicycles	9,9 billion	13,1 billion	14,4 billion
Passenger km by car	107,1 billion	141,6 billion	148,8 billion
No. of deceased cyclists	426	195	181
No. of deceased car occupants	910	477	371

All above-mentioned information about bicycle use in different countries reveals that the Netherlands takes a leading position in the bicycle traffic and infrastructure development now. The evidence of this fact is the highest bicycle share in all journeys, the highest kilometers cycled per inhabitant per day in Europe and others. Also, it may tell us that the most progressive ideas in the issue of bicycle policy and research originate from this country.

3. Cycling cities

The national average values of cycling parameters don't show the real situation in different cities of each country. The Figure 2.6 presents bicycle share in all journeys for several cities in different countries. For example, the level of cycling in the most bicycle-oriented places in the United Kingdom, Australia, Canada and the USA is lower than in the cities with the smallest value of bike mode share in the Netherlands, Germany and Denmark. The difference between these two groups is quite large. For instance, on the one hand, only in Germany there are few cities with bicycle use rate lower than 5 %. On the other hand, the value of the same parameter is much lower than 5 % in the majority of British and all American, Canadian and Australian cities.

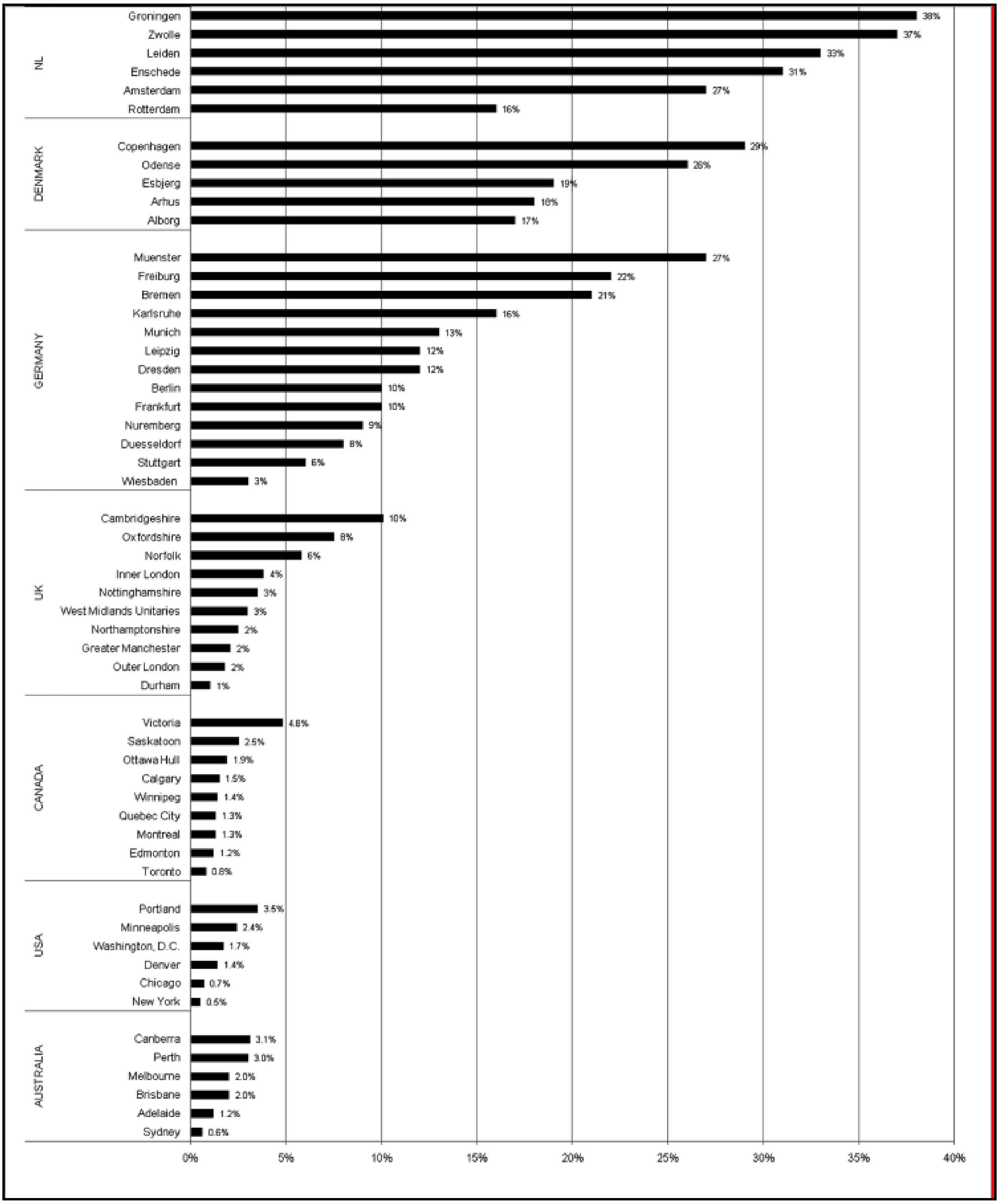


Figure 2.6 – Bike share of trips in selected cities in the Netherlands, Denmark, Germany, the UK, Canada, the USA and Australia (2000–2005). Note: UK data are for counties. (source: Pucher, John and Buehler, Ralph (2008))

In the table 2.2 ten cities with a respectable bike mode share in Europe are presented. The Dutch ones take the first positions in this list and have also higher level of bicycle use in comparison with the top "bicycle" cities of the other countries.

Table 2.2 – Bicycle use in ten European "cycling cities" (source: Fietsberaad Publication № 7)

Cities	Inhabitants	bicycle % in all inhabitants' trips
Groningen	182.000	37%
Zwolle	116.000	36%
Münster	272.000	36%
Veenendaal	62.000	32%
Kopenhagen	502.000	32%
Enschede	154.000	31%
Amsterdam	747.000	28%
Odense	187.000	26%
Freiburg (im Breisgau)	218.000	22%
Gent	237.000	17%

The Netherlands, Denmark, Germany and Belgium are the leaders in cycling in Europe. The Dutch top cities are Groningen and Zwolle, Veenendaal and Enschede are quite close to them. That's why the level of bicycle use in the "best cycling" cities of the Netherlands is completely different comparing with the cities not only in Germany, Denmark and Belgium, but in other European countries:

- Bicycle mode share in **the Netherlands** has been around 26% during the last years. The highest value of this parameter for some municipalities is between 30% and 45%, the lowest one is between 15% and 20%.
- The average bicycle share in **Denmark** is ca 20%. There is no big difference in bicycle use between the cities, and this number for urban areas remains around 20% as well.
- Cycling level in **Germany** is on average 10 %. The land of Nordrhein-Westfalen is known as German leader in the bicycle use. Looking at all German cities besides Münster and Freiburg there are ones which have relatively high bicycle share – between 20 % and 30 %.
- **Belgium** has the smallest bicycle mode share in comparison with all above-mentioned countries – 8%. The majority of Flanders' cities have almost reached the level of Ghent which is 15%. Some sources state that Bruges has a bicycle share around 20 %.

As it was written above, other European countries have quite low value of average bicycle use. But from time to time some extremes appear:

- **Great Britain** has an average bicycle use of 2%. But some isolated cities have high level of cycling: for example, York and Hull – 11%, Oxford and Cambridge – 20%. The same extremes can also be met in other countries, for example, Sweden and Italy.
- Bicycle use in **Ireland** is between 3% and 4%, whereas Dublin has 5%.
- **Sweden** has two values of cycling level: 7% as an average over the country and 10% for cities. As it was mentioned, there are some extremes as Lund and Malmö with bicycle use of 20%. Some sources name the city Vasteras (115,000 inhabitants) with incredibly high cycling level of 33%.
- **Czech Republic** doesn't stand out of other European countries. It would be better to say that there are problems with bicycle traffic as in the majority of Eastern European countries. An average bicycle use is below 5%. Of course, there are a number of cities with a sufficient cycling level: Ostrava, Olomouc and Ceske Budejovice have bicycle mode share between 5% and 10%. The Czech leader is the city Prostejov with cycling level of 20%.
- In **Austria** it is possible to extract two main leaders in bicycle use: Graz with 14% and Salzburg with 19%. An average value of cycling level is 9%.
- Similarly to Austria, **Switzerland** has a national bicycle mode share less than 10%. The cities with above the average value are Bern with 15%, Basel with 17% and Winterthur with around 20%.
- **France** has a quite low bicycle use rate of 5%. Only two cities stay apart: Strasbourg (12%) and Avignon (10%).
- The same national cycling level as France has another European country – **Italy**. But here it is also possible to find some extremes like Parma (over 15%) and Florence (over 20%). According to a number of sources the leader of Italy is Ferrara with the total bicycle use of 30%.

This subchapter describes and shows that bicycle use is quite different within Europe both on country and city levels. The level of Dutch cycling finds no comparison in Europe, even though Denmark has made a considerable progress in recent years. From a regional perspective the German region of Nordrhein-Westfalen and the Belgian region of Flanders must be mentioned as territories with a good bicycle traffic development. Another conclusion is that in the most non-bicycle European countries there is a number of cities with respectable cycling levels, such as in Great Britain, Sweden, Czech Republic and Italy.

4. The requirements for bicycle-friendly infrastructure

It is worthwhile to review the design rules of bicycle infrastructure that have led to such a rate of bicycle use in the Netherlands. There is a vast amount of literature on bicycle track and other bicycle facilities design. Many parameters and measure respect to national regulations. Here we shall give a short list of some important qualitative aspects that worth mentioning.

In order to create bicycle facilities, designers must pay special attention to the bicycle-cyclist system and the technical and physical properties of bicycle and cyclist. Here is a brief priority list of aspects that should be taken into account when designing a bicycle-friendly infrastructure:

- the required free space section must be guaranteed;
- two cyclist must have a possibility to ride side by side;
- the resistance when riding a bicycle must be minimized;
- physical and mental capacities and their limits must be taken into account (if it is possible, the mental capacity must be optimized);
- the cyclists' vulnerability must not be forgotten;
- the perception of cyclists must be taken into account;
- bicycle infrastructure must be complete and comprehensible [2, page 29].

The above-mentioned quality preferences can be transformed into 5 main requirements for a bicycle-friendly infrastructure:

- cyclists' perception and possibility to ride side by side return requirements in the field of *attractiveness* and *comfort*;
- the minimization of resistance when riding a bicycle establishes requirements in the field of *comfort* and *directness*;
- the section of free space and the optimization of mental capacity set up requirements in the field of *comfort* and *safety*;
- the cyclists' vulnerability establishes requirements in the field of *safety*;
- the necessity of a complete and comprehensible infrastructure for bicycle sets up requirements in the field of *cohesion* [2, page 30].

Various studies have shown that cycling infrastructure of a good quality actually leads to a higher proportion of bicycles in the modal split. One of the most recent studies in this area is 'Fietsbalans' (Bicycle Balance) which will be addressed in details in the next section.

5. The Cycle Balance

There is a number of different methodologies for cycling mobility research. The majority of them is very specific and can be implemented only within the scope of a certain project. The Cycle Balance is one of such projects the results, research methodologies and assessment approaches of which can be used in others bicycles and its mobility concerning studies. Assessment method of this project is also very closely connected with the main focus of the doctoral theses: the approach used in the Cycle Balance allows to describe and to evaluate bicycle traffic and bicycle infrastructure of different cities.

As it can be a good practical example and orienting point, the Cycle Balance will be considered in details below.

1.1. History and aims

The Dutch Cyclists' Union (*Fietsersbond*) presents and supports the interests of cyclists in the Netherlands, also makes a lot of efforts by creating and achieving better conditions for cycling and cyclists. It strives to increase competitiveness of bicycles in comparison with other modes of transport, especially, over short distances not only in terms of time, but also in terms of safety and comfort. To reach this aim it is necessary to develop a good bicycle-friendly infrastructure the main measures of which are directness, attractiveness, comfort, safe route and a coherent cycling network.

The Netherlands has the highest level of bicycle use among all European countries. That's why Dutch government, private companies and organizations support cycling by investing in it a lot of money and time. For example, Dutch bicycle infrastructure includes over 20 000 km of bicycle lanes and paths along the roads and almost 300 000 bicycle parking places next to the railway stations. Unfortunately, a quantitative assessment of their effectiveness and efficiency has never been done. To fill this gap the Dutch Cyclists' Union created, developed and executed the Cycle Balance (*Fietsbalans*). The Ministry of Transport, Public Works and Water Management has financed this benchmarking project. Local cycling conditions must be assessed impartially. The results are to be used to create a strategy which must help to improve local bicycle policies.

The project started in the summer of 1999 had two main objectives:

- The primary goal of the project was to motivate local authorities to study benchmarking techniques and then to use them for better cycling policy adoption. The idea of benchmarking is to learn some good practice from others: the results of the same action carried out for different study objects must be compared and the best one has to be chosen.

- The second goal of the Cycle Balance was to strengthen the position of the local *Fietsersbond* branches. The majority of decisions concerning cycling and its conditions is made on a local level. That is why it was important for local administrations and local governments to accept the *Fietsersbond* local branches as equal partners with enough knowledge and competence to represent the interests of cyclists. Thereby, the project was used to create a collaborative environment which could help to make deliberate decisions.

In 2002 115 towns participated in the project. This number included all Dutch cities with population over 100 000 people. Thus, more than half of all cyclists was covered.

1.2. The four surveys of the Cycle Balance

As it was written above, an impartial assessment of the local conditions for cyclists had to be made. To carry this out 10 different dimensions (also 24 sub-dimensions) of these conditions were determined. All dimensions are shown in the table 2.3. They provided a good mixture of major policy components including results, effects and process.

Table 2.3 – The ten dimensions of assessment in the Cycle Balance [21]

1. Directness	6. Bicycle use
2. Comfort (obstruction)	7. Road safety of cyclists
3. Comfort (road surface)	8. Urban density
4. Attractiveness	9. Cyclists satisfaction
5. Competitiveness compared to the car	10. Cycling policy on paper

One of the main ideas of the project was to use existing data if it was possible. All local information which is comparable, reliable and relevant could be a part of these data. Unfortunately, there is a small amount of existing databases which correspond to the mentioned conditions. For this reason within the project a special team was responsible for the necessary data collection. Thus, the research part of the Bicycle Balance included 4 surveys:

- A *questionnaire for municipalities* carried out with the aim to estimate the cycling policy provided by local authorities. A good bicycle policy focuses on the bicyclist and therefore satisfies the necessary quality requirements. Topics covered by this questionnaire were the following: cycling police plans and papers, bicycle facilities and network, budgets and the council which was considered as employer.

- A *questionnaire on cyclists' satisfaction* which would help to assess how the cycling conditions fulfilled the daily requirements of cyclists. Here a cyclist had to be considered as a customer. The motivation to use a bicycle is higher when the cycling conditions satisfy their needs. The questions concerned cycling comfort, parking for bicycle, prevention of bicycle theft, road safety when cycling, etc.
- *National databases* of different organizations as the Institute for Road Safety Research and Statistics Netherlands had also some information concerning local cycling conditions. With the help of these data relevant assessment units on bicycle use, road safety and urban density were calculated. Innovative analysis methods were applied as much as possible.
- *Quick Scan Indicator for Cycling Infrastructure* was a survey which helped to make a qualitative assessment of the local bicycle infrastructure. This survey is the most modern and innovative one among the others and was developed by the Fietsersbond together with engineering companies.

  First of all, a special measuring bike was developed (Figure 2.7). This bicycle was equipped with a laptop and a video camera. During cycling the computer collected the following data: time, speed, distance, stops, vibrations, sounds, etc. The camera registered at the same time different parameters of infrastructure: type of road surface, profile of the road, intersection type, barriers and maneuvers. All gathered information was sent to a specific software on the laptop which analyzed them and returned an assessment of the 4 above-mentioned dimensions: comfort (obstruction), comfort (road surface), directness and attractiveness.

  Within the scope of the Cycle Balance a number of routes was selected using a fixed random sampling approach. There were minimum 12 and maximum 16 of such routes for each participated town in its center and around the center. The aim was to create a representative picture of cyclists' movement in and around the city. Each route looked like a closed trip: it started from a randomly chosen residential house, then went to an attractive for cyclists destination and finished at another house.

  Every selected route was examined with help of the measuring bike. Also in parallel with this bicycle a car was driven on the same route at the same time. This allowed to estimate the time and cost of a trip by using a motor vehicle and, consequently, also the competitiveness of bicycle use.

Figure 2.7 – Dutch bicycle infrastructure measurement bike [21]

1.3. The assessment of participating towns

A comprehensive report was written for each city according to the results of the assessment made within the survey. In every report a description of the local cycling conditions was given basing on 10 dimensions and 24 sub-dimensions. All results were also compared with:

- the standards (existing and developed);
- the average values of all 100 cities and cities which have approximately the same size;
- the best city according to the evaluation.

Thus, it became possible, on the one hand, to understand deeper all bicycle policy aspects for each city and, on the other hand, to compare the assessment results and cycling policy efficiency between comparable towns. Also the reports revealed the aspects which had to be improved urgently.

Another outcome was the average assessment results of all participating towns. It showed common relationship between the considered indicators and the city size. Table 2.4 presents this outcome: dimensions and sub-dimensions, standard values and intervals for each sub-dimension, the average values for different towns were assessed. For example, small towns had an advantage of attractiveness and directness over big towns. Big cities in their turn got the edge on small ones in the competitiveness of bicycles.

Table 2.4 – General overview of the Cycle Balance assessment results (year 2000)

Assessed (sub)dimension	Standard	Interval	Overall Average	Average big towns*	Average medium size towns**	Average small towns***
Directness						
Detour factor (ratio)	1,25	0,1	Mediocre	Mediocre	very good	Mediocre
Delay (sec/km)	16,5	10	Good	Mediocre	Good	very good
Actual cycling speed (km/h)	15,5	1	Mediocre	Mediocre	Mediocre	Good
Overall judgement directness			**Mediocre**	**Mediocre**	**Mediocre**	**Mediocre**
Comfort (obstruction)						
Chance of stopping (N/km)	0,75	0,5	Mediocre	Bad	Mediocre	Good
Slow cycling and walking (% of time)	7,5	5	Mediocre	Mediocre	Mediocre	Mediocre
Traffic-obstruction (v-Fv)	1,75	1,5	Mediocre	Mediocre	Mediocre	Mediocre
Infrastructural impediment (v-Fi)	0,75	0,5	Mediocre	Mediocre	Mediocre	Bad
No right of way (N/km)	2,5	1	Mediocre	Mediocre	Mediocre	Bad
Turning off(N/km)	2	0,5	Mediocre	Mediocre	Mediocre	Bad
Overall judgement comfort (obstruction)			**Mediocre**	**Bad**	**Mediocre**	**Mediocre**
Comfort (road surface)						
Hindrance caused by vibrations (v-Ft)	100	40	**Mediocre**	**Mediocre**	**Mediocre**	**Bad**
Attractiveness						
Noise pollution (v-Fg)	130	40	**Mediocre**	**Bad**	**Mediocre**	**Good**
Competitiveness						
Journey time ratio (ratio)	1	0,1	Good	Good	Mediocre	Mediocre
Journey bikes faster (% of journeys)	70	20	Mediocre	Mediocre	Bad	Mediocre
Costs per journey (cents)	100	30	Bad	Good	Bad	very bad
Overall judgement Competitiveness			**mediocre**	**Good**	**Mediocre**	**Bad**
Bicycle use						
Share in trips to 7.5 km (%)	43	4	**mediocre**	**Mediocre**	**Bad**	**Mediocre**
Road safety of cyclists						
Victims per 100 million cycle km (N)	14	4	**mediocre**	**Mediocre**	**Bad**	**Mediocre**
Urban density						
Adresses per square kilometre (N)			**mediocre**	**Good**	**Good**	**Mediocre**
Cyclists satisfaction						
Bicycle parking (% dissatisfied)	17,5	15	mediocre	Mediocre	Mediocre	Bad
Comfort (% dissatisfied)	17,5	15	mediocre	Mediocre	Mediocre	Mediocre
Road safety for cyclists (% dissatisfied)	17,5	15	mediocre	Mediocre	Mediocre	Mediocre
Social safety (% dissatisfied)	17,5	15	Good	Good	Good	Good
Approach to bicycle theft (% dissatisfied)	17,5	15	very bad	very bad	very bad	very bad
Municipality's cycling ambitions (% dissatisfied)	17,5	15	Good	Good	Good	Good
Report mark	7,25	0,5	mediocre	Mediocre	Mediocre	Mediocre
Overall judgement cyclists satisfaction			**mediocre**	**Mediocre**	**Mediocre**	**Mediocre**
Cycling policy on paper						
Policy papers and plans (N)	16	4	mediocre	Mediocre	Mediocre	Bad
Bicycle network (N)	13,5	4	mediocre	Mediocre	Mediocre	Mediocre
Bicycle parking (N)	14	3	Bad	Mediocre	Mediocre	Bad
Budgets (N)	4	1	Bad	Mediocre	Bad	Bad
Council as employer (N)	5	1	mediocre	Mediocre	Good	Mediocre
Overall judgement cycling policy on paper			**mediocre**	**Mediocre**	**Mediocre**	**Mediocre**

* big towns = more then 100.000 inhabitants
** medium size towns = 50.000 - 100.000 inhabitants
*** small towns = 20.000 - 50.000 inhabitants

Coming back to each individual city it must be mentioned that their reports described the significance of each dimension and sub-dimension, collected data, determination of standards and the assessment approach. The main part of each report included a view on the assessment results and also conclusions based on these results. In the end a number of recommendations on existing situation improvement were proposed in the perspective of each dimension.

An illustration of the assessment results for each city can be done with help of the central graph based on the report results, conclusions and recommendations. In the Figure 2.8 the central graph of Veenendaal is shown as an example. It presents all strong sides of this city which are following:

- Veenendaal is quite compact city with a lot of destinations which could be reached by bicycle;
- bicycle usage rate and cyclists' satisfaction are enough high;
- the road surface quality is good;
- cycling policy corresponds to the standards provided by *Sign up for the bike* (design manual for cycle friendly infrastructure published by CROW in the year 1993);
- existence of free bicycle parkings in the center of Veenendaal and near the railway station.

It is also possible to see all weaknesses of the city in the central graph. Problems of cycling in Veenendaal and their feasible solutions are presented on the table 2.5.

Table 2.5 – Weak points of cycling mobility and their possible solutions in the city Veenendaal

Weak points of cycling mobility	Solutions
1. the risk of accidents with cyclists is relatively high, especially, on busy intersections	Reduction of car speed on such kind of intersections, construction of roundabouts where possible
2. High number of detours provides problems with directness. The main reason of this issue is connected with a small number of railway crossings for bicycles.	An analysis of the origins-destinations matrix must be carried out to find short routes for bicycles.
3. The obstructions on the roads produce problems for bicycle traffic: uncomfortable travel and low average speed. The reasons are normally network design and other road users, especially, cars.	Motorized traffic must be prohibited on the residential roads.
4. There is a number of reasons which have influence on the competitiveness of the bicycle. The availability of quicker trips for cars and low cost of car parking are among them.	Designing of direct routes for bicycles, restriction of car traffic in the city center, increase of car parking costs in a number of important places of a town.

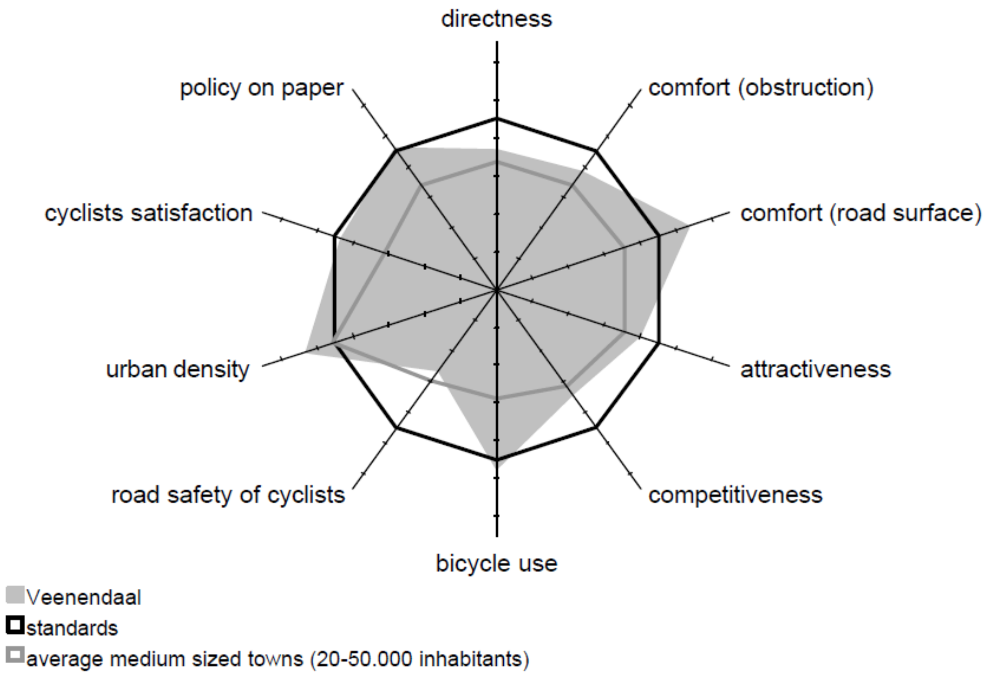


Figure 2.8 – The Cycle Balance score of Veenendaal

1.4. Good cycling policy works

All surveys carried out during the Bicycle Balance project created unique databases which contain relevant information about the cycling conditions in the 115 towns of the Netherlands. It was the first time when such amount of data were gathered using a systematic approach. Besides the reports were made for each city and analyses of the collected data were done on common rules. The main outcome of these analyses is a strong correlation found between the bicycle use and the values of the other 9 dimensions for bicycle traffic. The correlation graph presented in the Figure 2.9 shows that bicycle use in the cities with the high Cycle Balance score is 30 % higher than in the cities with a low score. It confirms the idea that the assessment method used in the project is meaningful. It means also that an effective bicycle policy yields favorable results in creation of good conditions for cycling and cyclists.

Though the correlation showed in the graph is quite strong, only 20 % of difference in bicycle usage among the participating cities can be explained by this correlation. To get a full picture some other influencing factors have to be considered:

- the use of public transport to travel for short trips;
- allowing public transport in the model as an influence factor on dimension “competitiveness of the bicycle” increases severalfold compared to other ones;
- characteristics of the city population.

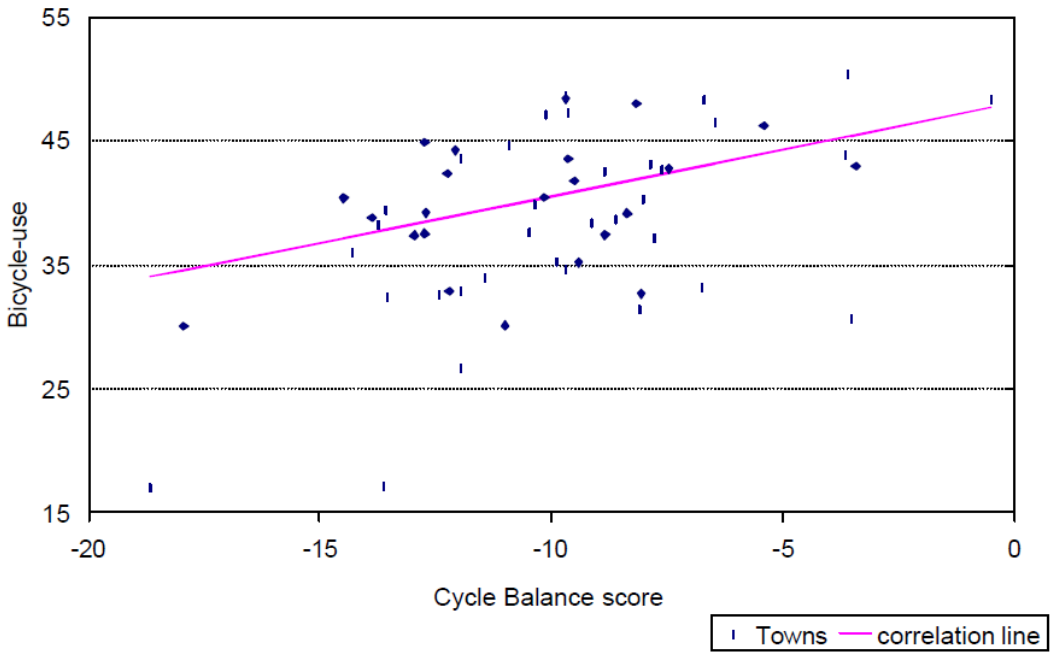


Figure 2.9 – Correlation between bicycle-use and score on the Cycle Balance

1.5. Publicity and "Cycle City" elections

One of the important aspects of the Bicycle Balance project was collaboration with mass media and its high attention to all activities. To influence the opinion of the society on all levels was an essential condition of the successful project realization. For example, the Quick Scan measurements were accompanied by a great media interest. Not only the process of data collection, but also results presentations and their public discussions were interesting events for both local and regional mass media.

Also a competition between the cities was organized. The idea was to find the best city according to the analysis and to give to this city the title of "Cycle-city 2000". At first, five challengers were chosen in November 2000. On the second step an independent commission selected the winner. It was Veenendaal which had the most coherent bicycle policy and the best bicycle infrastructure among the other candidates. The title "Cycle-city 2002" went to Groningen which had the biggest bicycle share in the modal spit (47%). An interest to this competition was used to organize communication meetings in the nominated cities where a lot of administration employees from other places were invited to learn the good practices.

Thus, the Bicycle Balance project became well-known for everyone who works with bicycle traffic and infrastructure.

6. Conclusion of Chapter 2

This chapter illustrates the historical development of cycling culture in European cities during the previous century and compares it with today's bicycle usage. It has been shown that a strong cycling history results in most cases in a high bicycle usage today, but there are also exceptions.

A number of European countries and cities were considered in the perspective of common cycling parameters including bicycle use and its differences. They help to assess the general situation with cycling mobility both on country and city levels. According to the assessment of all these parameters the Netherlands takes the leading position in all spheres of cycling: from the attitude to cyclists as road users and bicycle as a mode of transport until the development and implementation of bicycle policies.

The highest bicycle usage on national and city levels is in the Netherlands. Also its leadership in other cycling issues mean that this country has the most progressive ideas in the cycling research including the development of assessment and analysis methods for bicycle traffic, demand and infrastructure. Thereby, for example, basing on the Dutch advanced experience the requirements for a bicycle-friendly infrastructure were formulated.

Furthermore, the Netherlands have also developed and carried out the most sophisticated project concerning cycling research, the 'Fietsbalans' (Bicycle Balance), which was described in details. It included two steps:

1. data collection on bicycle traffic, infrastructure and policy;
2. assessment of the gathered information.

Among the outputs of the Bicycle Balance project there are reports on local cycling conditions for each participating city, the comparison and assessment results, unique database which includes all collected data and was different correlations. The selection of the "best cycling city" in the Netherlands can be also taken as a project output. Unfortunately, the exact calculation of the Bicycle Balance has not been published, but data and methods could be obtained from the Dutch Cyclists' Union (the majority of the manuals are in Dutch language).

Finally it must be mentioned that information about collected data, approaches used in each survey, assessment methods and common ideas of the above-described project had a positive influence on the achievement of the theses aims.

Chapter 3 – Automated Analysis Method for Bicycle Demand Assessment

1. Introcution

As it was mentioned above, automated analysis method for bicycle demand assessment were based on three components: *mobility survey*, *survey evaluation* and *potential demand assessment*. But mobility survey has the strongest influence on this method development as other two activities. It is possible to say that this survey does not influence only on its development, but shows the main directions of its design.

To explain above-written statement the steps of method development must be remembered. They were declared in the subchapter 5 of chapter 1. The first step requires understanding the necessity of two WP3 components: *survey evaluation* and *potential demand assessment*. In the perspective of the method development these components return the main outputs concerning assessment of existing and potential demand of bicycle usage. That means their importance for approach design. But it must not forget that they (as also indicators calculation) use data getting by *mobility survey*. By comparing all three activities the most labor-consuming one is *mobility survey*. It takes a lot of time not only by survey organization, but also by data obtaining from questionnaires and other data carriers.

Thus, the main orienting point for creation and development of automated analysis method is mobility survey. But aslo it is necessary to remember about its close connection with subsequent phases (*survey evaluation* and *potential demand assessment*). Their relationship gives information about outcomes planned after data evaluation. It must be taken into account by design of survey in common and of questionnaires particularly.

2. Survey, its development and types

A survey is a data collection tool used to gather information about individuals. Surveys are commonly used in psychology research to collect self-report data from study participants. A survey may focus on factual information about individuals, or it might aim to collect the opinions of the survey takers.

Surveys are generally standardized to ensure that they have reliability and validity. Standardization is also important so that the results can be generalized to the larger population.

2.1. Types of Survey

Surveys can be implemented in a number of different ways. Chances are good that you have participated in a number of different market research surveys in the past.

Some of the most common ways to administer survey include:

- Structured interview – the researcher asks each participant the questions.
- Questionnaire – the participant fills out the survey on his or her own.
- Mail - An example might include an alumni survey distributed via direct mail by your alma mater.
- Telephone - An example of a telephone survey would be a market research call about your experiences with a certain consumer product.
- Online - Online surveys might focus on your experience with a particular retailer, product or website.
- At home interviews - The U.S. Census is a good example of an at-home interview survey administration.

2.2. Advantages and Disadvantages of Surveys Using

As every data collection method survey has its own advantages and disadavantages.

The following points can be counted as advantages:

- Surveys allow researchers to collect a large amount of data in a relatively short period of time.
- Surveys are less expensive than many other data collection techniques.
- Surveys can be created quickly and administered easily.
- Surveys can be used to collect information on a wide range of things, including personal facts, attitudes, past behaviors and opinions.

Also disadvantages can be named:

- Poor survey construction and administration can undermine otherwise well-designed studies.

- The answer choices provided on a survey may not be an accurate reflection of how the participants truly feels.
- While random sampling is generally used to select participants, response rates can bias the results of a survey.

3. Mobility Survey of the BICY project

The mobility survey has the objective to collect all transport-user related data in a unified manner. The survey type that would meet best all objectives, including the budget constraints, is a supervised street-survey (household interview survey would be too costly, a phone interview may be less expensive, but it is less reliable). With a street survey the partners would need to distribute and collect the questionnaires at specific places or events.

The detailed mobility survey was designed to complement, cross-validate, and build upon the Common Indicators (described above).

The survey affords an important snapshot in time of both the *travel behaviour* and *individuals' experiences* in each Partner Place, and also provides prospective data as to how people's choices would change, given new options. Respondents were asked how their travel choices would change given a variety of different conditions such as improved bikeways, and improved public transport options.

From this data, along with the indicator data, a survey evaluation will be done and bicycle demand will be assessed.

Details regarding the survey, the data obtained, and its methodology will be considerd below.

3.1. Aim of Moility Survey

The survey was conducted to find out which obstacles to cycling different cities in Europe have and under what circumstances they would consider using the bike also which obstacles are common and which are specific to the region. The point of this data collecting was also to analyze the data so trans-national strategies could be formed and since there were not any comparable data available in different cities in Europe conducting a survey was the only option to acquire required data. But why conduct a street-survey? The answer is simple. Since the aim was to get adequate comparable data the survey had to involve cyclists and non-cyclists but also to get as many questionnaires filled out as possible in a short time so conducting the survey would be efficient. But to get good data the questionnaire had to be understandable for everybody which means it had to be in their native language. It was also important to choose a good location for the sampling site to get un-biased results and the site had to be representative for the population. The staff who conducted the survey had to ensure smooth distribution and collecting the questionnaires. They also had to make sure that it was impossible for somebody to fill out many questionnaires, that was prevented by the fact that people who filled out the forms handed them in one by one and that way if there were any questions the staff could assist them so rate of perfectly filled questionnaires would rise.

3.2. Localization and adaptioan of questionnaires

All the questionnaires were localized which means it was translated into native language where the survey was conducted (if it was not already available) and it was necessary to adapt some other data too. If the localization was done it was sent back to Unibo before proceeding. Basically the steps that was taken to localize the questionnaires were:

1) It had to be taken into consideration that the translated questions had exactly the same meaning.
2) Distance home residence: they inserted the name of the city and possibly the place/street which was considered the center.
3) The residence block of the survey was adapted. That involved filling out the cities name where the survey was done in then the adjustment in the distances between city center and residential area. That was done to roughly determine the urban density which was necessary for comparing different cities. Some rules of thumb could be the following:
    - The first distance should be the inner city or the city center which usually has the highest population density
    - The largest distance shoulnt be farther from the center than city limits
    - The survey boxes can be placed in every city quarter unless it is a small city and the distances would be lass than 500 m

3.3. Choice of survey sites and budgeting

The survey sites were chosen carefully. That means that the site for conducting survey needed to be a place where a lot of people were. Usually something like city center or the area with highest population density. In order to get adequate results from the survey the staff who carried out the survey needed to get around 1500 filled questionnaires. Actually the number depends on the size of the city or town the survey is carried out. It is also very important to involve representative population of the town/city in the survey to get un-biased results. Now the budgeting will determine how much money is available. The survey conducted as all the future surveys use(d) survey calculator which makes calculating the costs of this survey really easy as only few parameters had to be changed to see approximately what the costs of conducting this is. The calculator involves everything that needs to be considered. For example the surveys that were conducted filled in the number of personnel, how long the event is(how many days and how many hours per day), and also the cost of the person per hour, gadget cost and cost of questionnaire and the calculator basically calculated all the costs that can be expected and even gave the estimated number of questionnaires that needs to be printed out of course the number of reserve questionnaires had to be given by the person who entered the data to the calculator. So the basic aim of this calculator was to give an idea how much it costs to conduct the survey.

3.4. Preparations and carrying out the survey event

The preparations for the survey were actually easy as it was needed to print the questionnaires and procure the T-shirts with Central Europe logo and procure the gadgets. There was also need for chairs and tables where people can fill out the questionnaires and if the event was outside then umbrellas or tents were necessary. It is also necessary to train the staff who conduct the survey as they need to remember the questionnaire and if needed help the person who is filling out the form. Conducting the survey turned out to be most effective if each trained assistant, equipped with a questionnaire, is approaching individually the interviewee and fills in the questionnaire together with him/her. If the interest in this survey is greater than expected then assistants number should be increased to guarantee smooth process. But some younger interviewees wanted to fill out the form by themselves and in this case some control questions should be asked just to make sure the answers are correct.

Figure 3.1 – Carrying out an interview

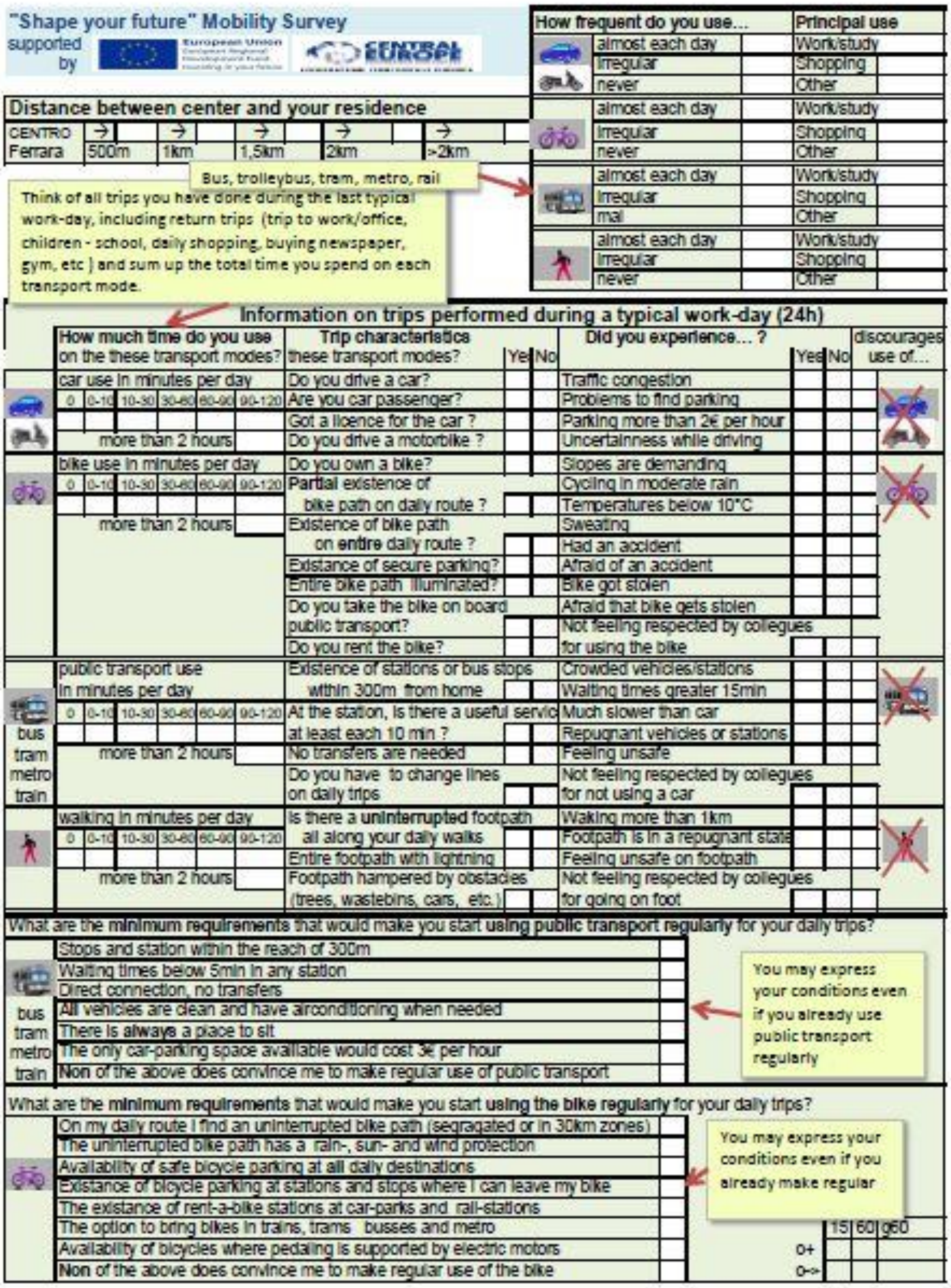


Figure 3.2 – The one page mobility survey

3.5. Questionnaire design/ Detailed instruction for questionnaire compiling

Filling out the survey is easy as people had to mark the right option with the x. It was also possible to have multiple answers unless the questions were yes or no. Yes and no questions had 2 boxes to put the x in the first box was for "yes" and the second one for "no". The survey itself consisted different blocks of questions. For example residence block which contains the cities name and where this person lives in this city for example if he/she lives in the city center or outside of it and the distance to the center. The next block was general mobility. In this block the questions were about different transport modes and how often the interviewee uses it and with what purpose for example for going to work, shopping or for other reason. Then came the largest block it was about regular work day and the questions in there where about how many minutes per day a interviewee uses different modes of transport. It also involved questions about infrastructure characteristics and personal experiences on their everyday trips. This block had to be filled out about interviewees most typical day. The future transport block which contained questions about circumstances a person would use public transport or bike regularly. Of course it also had the opinion that the person would never use this mode of transport. And now the last but not the least the questions about the age and sex of the interviewee.

4. Surveymaster

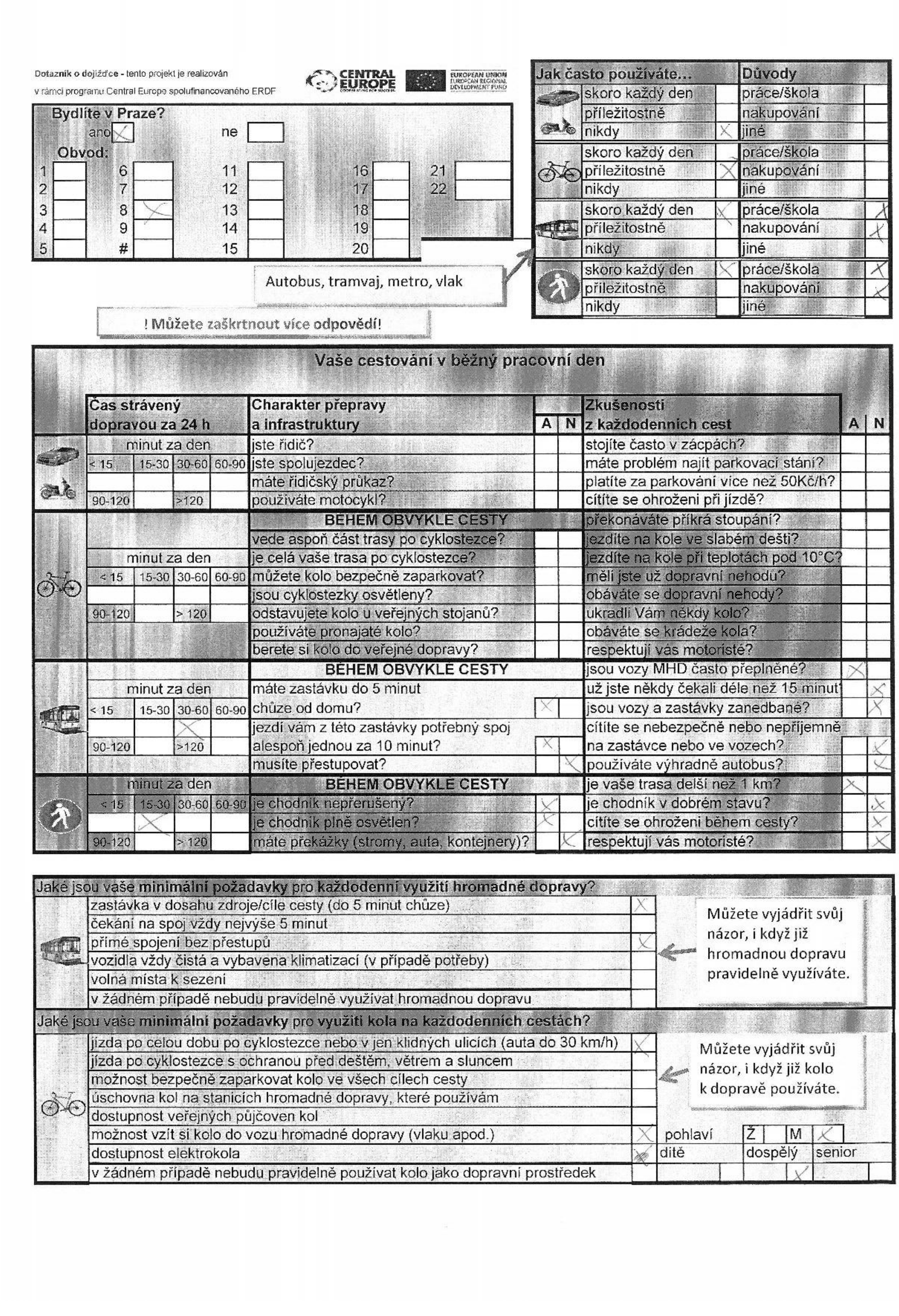
In the end there were 13 cities from 7 different countries which participated in this mobility survey. The main output of thess street surveys was over 9000 questionnaires. Each questionnaire had only one page, but it did not reduce an amount of manual work.

By this reason special software “Surveymaster” was created. In this subchapter this software, its description and application within the scope of the BICY project will be considered.

4.1. Scanning of questionnaire

This action can be regarded as pre-stage before using the “Surveymaster”. The aim of this activity follows from its name: to scan the whole questionnaires. There are also a number of requirements for this process:

- scanning must be done by using a white-black format;
- questionnaires can be scanned together (in group of 25-30 pieces);
- scanned questionnaires must be saved in PDF-format.



Dotazník o dojížďce - tento projekt je realizován v rámci programu Central Europe spolufinancovaného ERDF

CENTRAL EUROPE — EUROPEAN UNION EUROPEAN REGIONAL DEVELOPMENT FUND

Bydlíte v Praze? ano ☒ ne ☐

Obvod: 1 2 3 4 5 6 7 8 9 # 11 12 13 14 15 16 17 18 19 20 21 22

Jak často používáte...		Důvody	
skoro každý den		práce/škola	
příležitostně		nakupování	
nikdy	X	jiné	
skoro každý den		práce/škola	
příležitostně	X	nakupování	
nikdy		jiné	
skoro každý den	X	práce/škola	X
příležitostně		nakupování	X
nikdy		jiné	
skoro každý den	X	práce/škola	X
příležitostně		nakupování	X
nikdy		jiné	

Autobus, tramvaj, metro, vlak

! Můžete zaškrtnout více odpovědí!

Vaše cestování v běžný pracovní den

Čas strávený dopravou za 24 h	Charakter přepravy a infrastruktury	A	N	Zkušenosti z každodenních cest	A	N
minut za den	jste řidič?			stojíte často v zácpách?		
< 15, 15-30, 30-60, 60-90	jste spolujezdec?			máte problém najít parkovací stání?		
	máte řidičský průkaz?			platíte za parkování více než 50Kč/h?		
90-120, >120	používáte motocykl?			cítíte se ohroženi při jízdě?		
	BĚHEM OBVYKLÉ CESTY			překonáváte příkrá stoupání?		
	vede aspoň část trasy po cyklostezce?			jezdíte na kole ve slabém dešti?		
minut za den	je celá vaše trasa po cyklostezce?			jezdíte na kole při teplotách pod 10°C?		
< 15, 15-30, 30-60, 60-90	můžete kolo bezpečně zaparkovat?			měli jste už dopravní nehodu?		
	jsou cyklostezky osvětleny?			obáváte se dopravní nehody?		
90-120, > 120	odstavujete kolo u veřejných stojanů?			ukradli Vám někdy kolo?		
	používáte pronajaté kolo?			obáváte se krádeže kola?		
	berete si kolo do veřejné dopravy?			respektují vás motoristé?		
	BĚHEM OBVYKLÉ CESTY			jsou vozy MHD často přeplněné?	X	
minut za den	máte zastávku do 5 minut			už jste někdy čekali déle než 15 minut?		X
< 15, 15-30, 30-60 (X), 60-90	chůze od domu?	X		jsou vozy a zastávky zanedbané?		X
	jezdí vám z této zastávky potřebný spoj			cítíte se nebezpečně nebo nepříjemně		
90-120, >120	alespoň jednou za 10 minut?	X		na zastávce nebo ve vozech?		X
	musíte přestupovat?		X	používáte výhradně autobus?		X
minut za den	BĚHEM OBVYKLÉ CESTY			je vaše trasa delší než 1 km?	X	
< 15 (X), 15-30, 30-60, 60-90	je chodník nepřerušený?	X		je chodník v dobrém stavu?		X
	je chodník plně osvětlen?	X		cítíte se ohroženi během cesty?		X
90-120, > 120	máte překážky (stromy, auta, kontejnery)?		X	respektují vás motoristé?		X

Jaké jsou vaše **minimální požadavky** pro **každodenní využití hromadné dopravy**?

zastávka v dosahu zdroje/cíle cesty (do 5 minut chůze)	X
čekání na spoj vždy nejvýše 5 minut	
přímé spojení bez přestupů	X
vozidla vždy čistá a vybavena klimatizací (v případě potřeby)	
volná místa k sezení	
v žádném případě nebudu pravidelně využívat hromadnou dopravu	

Můžete vyjádřit svůj názor, i když již hromadnou dopravu pravidelně využíváte.

Jaké jsou vaše **minimální požadavky** pro **využití kola na každodenních cestách**?

jízda po celou dobu po cyklostezce nebo v jen klidných ulicích (auta do 30 km/h)	X
jízda po cyklostezce s ochranou před deštěm, větrem a sluncem	
možnost bezpečně zaparkovat kolo ve všech cílech cesty	
úschovna kol na stanicích hromadné dopravy, které používám	
dostupnost veřejných půjčoven kol	
možnost vzít si kolo do vozu hromadné dopravy (vlaku apod.)	X
dostupnost elektrokola	X
v žádném případě nebudu pravidelně používat kolo jako dopravní prostředek	

Můžete vyjádřit svůj názor, i když již kolo k dopravě používáte.

pohlaví	Ž		M	X
dítě		dospělý		senior
			X	

Figure 3.3 – Sample of questionnaire

4.2. Surveymaster interface

The manual is a description how to work with the program Surveymaster developed by UNIBO-DICAM team.

This program has the following aims:

- to make a processing of questionnaires automated which minimizes time expenses

- on the one hand, to prevent human failures using automated processing method which minimizes an influence of human factor on the quality of results (especially, by a large amount of survey materials)
- on the other hand, to provide for a required minimum human control by working with software which helps to avoid errors coming with process automation and, as a consequence, to increase the quality of output

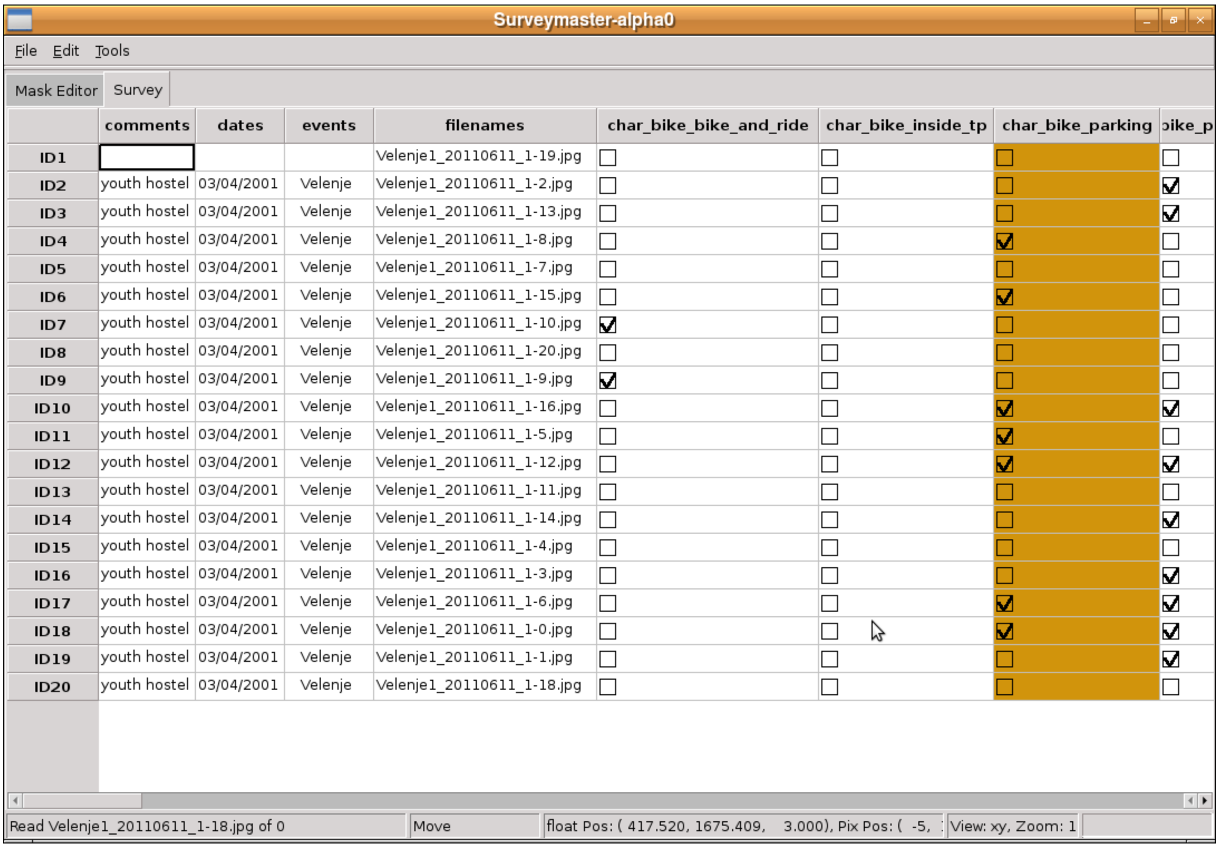


Figure 3.4 – Surveymaster interface

5. Conclusions of Chapter 3

Creation and development of automated analysis method for bicycle demand assessment was the main focus of this chapter.

The survey was designed to complement, cross-validate, and build upon the Indicator Data (described above), for use in transnational analyses.

The survey affords an important snapshot in time of both the *travel behaviour* and *individuals' experiences* in each Partner Place, and also provides prospective data as to how people's choices would change, given new options. Respondents were asked how their travel choices would change given a variety of different conditions such as improved bikeways, and improved public transport options.

From this data, along with the indicator data, the potential success of interventions is evaluated, including carbon reductions and health-related cost-benefit analyses.

Chapter 4 – Automated Analysis Method for Bicycle Infrastructure Assessment

1. Introduction

Creation and development of the automated analysis method for bicycle infrastructure assessment was one of the two main goals of these theses.

As it was mentioned above, the basis for this method is a Common Indicators activity of the BICY project and, especially, of its three elements: *infrastructure data collection, infrastructure analysis* and *indicators calculation*. According to these three elements the process includes the following stages:

- The first stage focuses on *infrastructure data collection* and has an aim to determine the source which will provide full, comparable and available data on different types of infrastructure.
- At the second stage *indicators calculation* is carried out with the aim to find out what information is necessary and relevant for bicycle infrastructure assessment.
- The third stage deals with *infrastructure analysis* aiming at establishment of the collected infrastructure data analysis schemes in order to get suitable outputs mentioned on the second stage.

This analysis method was realized in the form of special software based on the Python programming language. It was done to make this approach really automated.

All three development stages of the method and, correspondingly, the type of software used will be considered in details in this chapter.

1. OpenStreetMap as data source

There was a number of problems met while infrastructure data collection. The main obstacle was that necessary spatial data received from different cities and regions were inconsistent or incomplete. This has stimulated to search through all alternative sources of data which would correspond to the required conditions.

After considering various opportunities the collaborative project OpenStreetMap (OSM) was selected. The aim of this project is to design and develop a free editable map which covers the whole Earth. In other words, OSM is a database with a huge amount of free data which is being constantly updated by users all over the world and can be downloaded fast and easily.

The main information concerning OpenStreetMap and some aspects of working with it will be presented below in the next subchapter.

1.1. History and development of OpenStreetMap

According to founders of OpenStreetMap, most of free maps have technical or legal restrictions on use. That is why they invented and developed OpenStreetMap, a collaborative project to create a free editable map of the whole Earth. OSM geographical data is free to anyone who is interested in them and every registered user can change and improve the map content. Data from GPS devices, aerial photography, video, satellite images, panoramas of streets and other sources is used to create the OSM.

The idea of creating the OpenStreetMap belongs to Steve Coast. The project was founded in 2004 in the United Kingdom. Its main aim was firstly to provide a free access to the full map dataset, because most of so-called free maps have limited access to data in reality. The “openstreetmap.org” domain name was registered in August of the same year.

In January 2006 the offline editor JOSM was released and soon it became the most popular editor to make changes in the map content. In the same year the non-profit organization named the OpenStreetMap Foundation was established. Its main function was to support the development of free spatial data and its distribution and also to give a possibility for everyone to use and share information. The OpenStreetMap Foundation has a rich membership which brings together people from different countries.

In July 2007 the first OSM international The State of the Map conference was held, which has such sponsors as Yahoo, Google and Multimap. The first organization that began to use OpenStreetMap data was Oxford University. It happened in December 2007.

Venture financing in the sum of 2.4 million Euros was received in March 2008 from CloudMade company which uses the data from OSM. OpenStreetMap has grown to 978, 483 contributors during 8 years (this information was actual for date 20.12.2012).

Theoretically, OpenStreetMap covers the whole Earth, but some countries have full datasets (like Germany) and others only location of main roads and settlements (Somalia, Chad, North Korea).

The Figure 4.1 shows the statistics on OpenStreetMap use (February 2011).

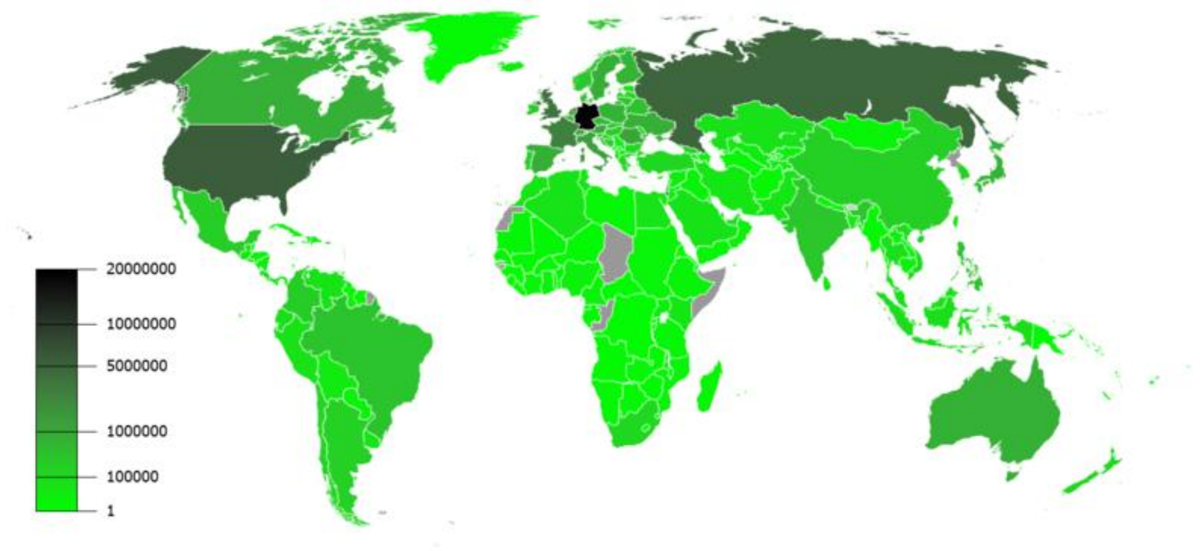


Figure 4.1 – The statistics of OpenStreetMap usage by the end of February 2011

It can be concluded that countries, where people use the OSM actively, have more detailed maps than countries with small percentage of users.

The main site, where OSM is located, is http://www.openstreetmap.org/. It contains the map, which is editable for registered users, the history of corrections and opportunity to export data. The map has several views: Standard, Cycle Map, Transport Map and MapQuest Open.

The next important site that describes the OSM project is http://wiki.openstreetmap.org/wiki/Main_Page. Firstly, this website references on the main online map. Also it contains different descriptions related to OSM: information about different mapping projects, Beginner`s guide to learn how a person can improve geographical data, Map Features, Developers and Licensing.

The full information about Open Street Map can find in Wikipedia: http://en.wikipedia.org/wiki/OpenStreetMap. There is a OSM history, contributors, map production, licensing, use of OpenStreetMap, software and data format on the site.

1.2. OSM map production

The OSM gets data from different sources. The first data was obtained by developers on a voluntary basis. Map data collectors moved on foot, by car, bicycle or using other modes of transport. They used different data recorders such as digital camera and GPS. After that all tracks were exported into the OpenStreetMap database.

Later, some government agencies and commercial organizations have made available works that contain street data and satellite imagery sources. Aerial photography allows drawing maps of major cities without tracks and consequently simplifies maps construction.

The main government services, which are used by the OSM, are Landsat, Prototype Global Shorelines (PGS), TIGER and cartographic services – Bing Maps.

Every person can use this map, export the data and see the history of corrections, but only registered users can be editors and create and share map information.

A website www.learnosm.org/ is source which instructs how to register own account and make maps with OSM.

The registration occurs on OSM website www.openstreetmap.org. It is necessary to fill the following boxes: email address, username and password to register an account. After registration user should open his/her email to confirm created account by clicking appropriate link. Each registered user gets user identification number or briefly *uid*. It can be seen after editing in XML-file of OSM data how is shown in the code of Figure 4.2 (with bold type). Construction of OSM XML-file will be considered in details below.


```
<osm version="0.6" generator="OpenStreetMap server" copyright="OpenStreetMap and
contributors" attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
<changeset id="14538919" user="OSMsvetlana" uid="1156757" created_at="2013-01-
05T16:40:30Z" open="true" min_lat="59.3938775" min_lon="27.770864" max_lat="59.3938775"
max_lon="27.770864">
<tag k="comment" v="Swimming pool was added in Sillamäe, Estonia (local knowledge)"/>
<tag k="version" v="2.3"/>
<tag k="created_by" v="Potlatch 2"/>
<tag k="build" v="2.3-554-ge648197"/>
</changeset>
</osm>
```


Figure 4.2 – Example of XML-file of OpenStreetMap data (user identification number is highlighted with bold type)

OpenStreetMap has different programmes called OSM Editors to edit maps:

- *JOSM* – offline Java OpenStreetMap editor. It is more suitable for a skilled map constructor. JOSM will be considered below.
- *Potlatch 2* – is an OpenStreetMap editor that allows to make edits directly through the website www.openstreetmap.org. It is the easiest way for beginners and it was rewritten from the Potlatch.
- *OSM2go* is an editor for mobile devices and desktop computers. This editor includes opportunities for manipulation with ways, nodes and their tags. It is also possible to import GPS traces and to manage multiple mapping projects.

- *Merkatoor* is an OpenStreetMap editor which can be used on the following operating systems: Unix, Windows and Mac OS X. Merkatoor is one of the most popular editors after JOSM Potlatch 2.
- *Mapzen* is a set of tools for editing maps. It includes Flash-based online editor and an iPhone and Android application for adding and editing points of interest by users.
- *Vespucci* is the first OpenStreetMap editor designed for mobile devices that work on the Android.

As it has been shown above the OSM has a lot of editors, but the most popular are Potlatch and JOSM. The key difference between them is that the first one is online and the second one is offline editor. Within the scope of this work only JOSM will be presented more precisely.

As it was mentioned above, JOSM is offline OpenStreetMap editor. It is fully functional and suitable because of:

- possibility to work with big data;
- availability of photos done during the track recording and if desired which could be presented on the map;
- possibility to editing the area which is being processed by another users.

JOSM requires installation on the computer and, at the same time, it is necessary to have Java 1.5 or later version. This editor supports loading of data from different formats. Between them GPX can be found which is a text format based on XML for storing and exchanging GPS data.

To start working with JOSM a user must open this editor on the personal computer after its installation. Then a data from OpenStreetMap server must be chosen and downloaded. With this data user can do any further manipulations. An example of JOSM editor interface is shown in the Figure 4.3.

In the centre of the window there is an editable map and on the left the main panel with buttons. The map contains points or nodes (for example, a shop), lines or ways (e.g. a road) and shapes of enclosed filled area (for example, buildings, forest, river).

The main panel consists of a number of buttons. The first four buttons are the most important when using JOSM. They are “Select”, “Draw”, “Zoom in” and “Delete”.

To draw a point a user should choose “Draw” button and click twice on the map. For drawing lines user should make single-click and click on the each subsequent point of the object until the line is ready. The same principle is used for drawing shapes, but double-click must be done on the start point.

To change the object’s location, a user should select the object by clicking on the line around it. Then it must be dragged with the mouse to a new place on the map.

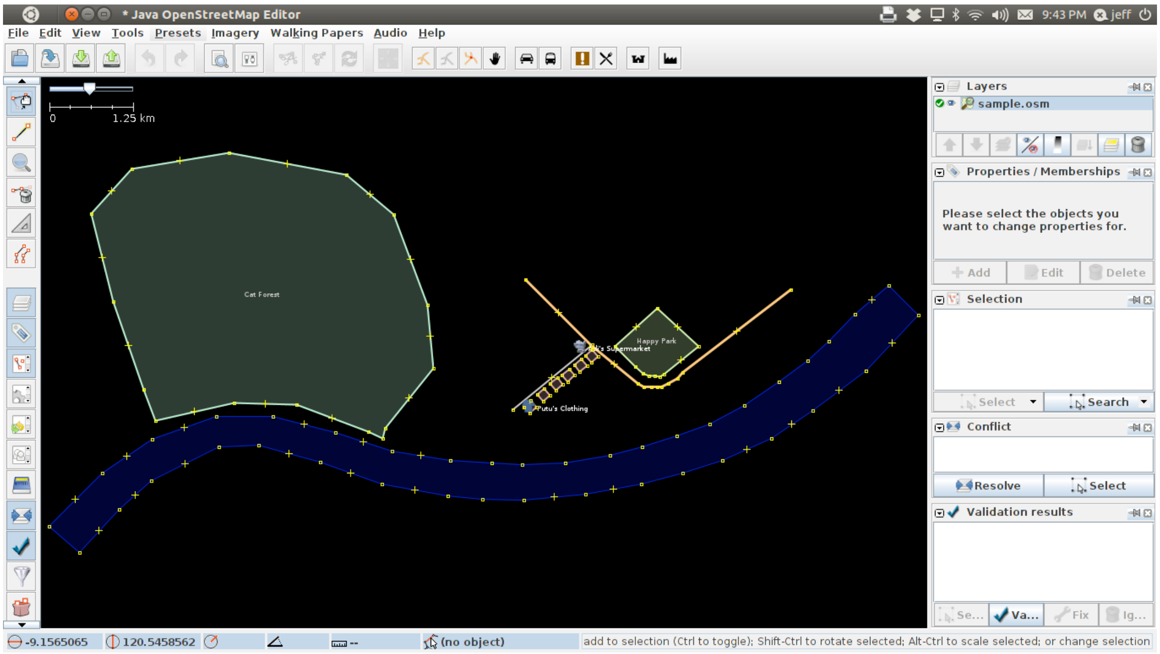


Figure 4.3 – Interface of JOSM Editor

It is enough to select a single-point and to drag it with the mouse for changing the object's shape. When using the OpenStreetMaps, adding and editing them with editors, different questions can arise. In order to discuss various problems, the main OSM forum was opened for users. Its link is http://forum.openstreetmap.org/. The main topics are divided into five groups: OpenStreetMap, OSM Specials, OSM Forum, OSM Community and Archive.

The first group considers the following topics: Questions and Answers related to OSM, Editors (JOSM, Potlatch etc.), Development, Meetings and Events. Each user can find and seek for necessary information there.

OSM Forum consists of general chat and feedback part. Moreover, it was a good idea to divide OSM community into different countries. It simplifies communication between people, who live in the same country, because users who know about their region better than others can communicate in their language.

1.3. OSM data structure

OpenStreetMap has the following elements or so-called data primitives: *NODE*, *WAY*, *RELATION*.

A *NODE* defines a single point. Its symbol in the OSM is shown in the Figure 4.4. In comparison with other primary elements, a NODE has always coordinates features: longitude and latitude (sometimes, there is also altitude). Besides the common attributes and tags which will be discussed below.

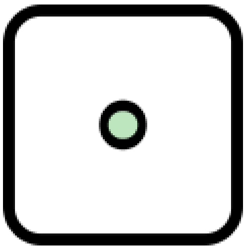


Figure 4.4 – OSM primary element NODE

Nodes can be used to define:

- a single point with its own features or
- a point which will be the part of a way.

In the first case a node must have at least one tag. A tag describes elements such as nodes, ways and relations. It is pair of "Key" and "Value". Keys are unique, they don't occur twice for one element. Term "tag" will be considered more in details below.

A *WAY* is an ordered list of nodes connected by edges between each others. Every WAY has at least one tag. The number of nodes in one way can be from 2 to 2000. The *WAY* is divided into 3 types:

1. *Open polyline* represents a linear feature. For example, a road is an open polyline. OSM symbol of this WAY type is shown in the Figure 4.5.

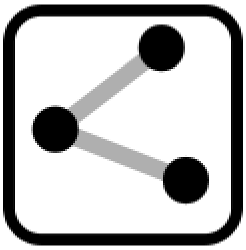


Figure 4.5 – Open polyline as type of OSM primary element WAY

2. *Closed polyline* is a way the first and last nodes of which are the same. For example, the walls which go around the entire perimeter of a property. OSM symbol of this WAY type is shown in the Figure 4.6.

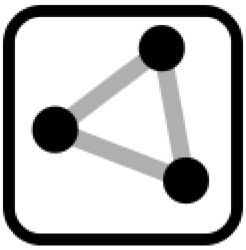


Figure 4.6 – Closed polyline as type of OSM primary element WAY

3. An *area* is closed WAY which is used to define a certain territory as an enclosed filled area. An example of such WAY type can be the perimeter of forest. OSM symbol of this WAY type is shown in the Figure 4.7.

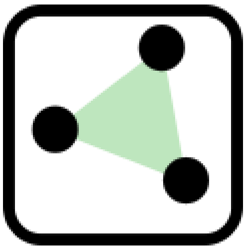


Figure 4.7 – Area as type of OSM primary element WAY

RELATION is an ordered list of nodes and/or ways which has at least one tag. OSM symbol of element RELATION is shown in the Figure 4.8. This element consists of members, which are formed by roles and nodes, ways or relation. A role describes which role a node, way or relation has.

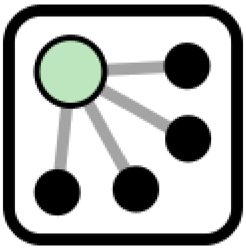


Figure 4.8 – OSM primary element RELATION

Every element of data primitives has a number of characteristics which describe them and also make them unique in comparison with the other elements. It is necessary, for example, to make difference between two nodes, or two ways, or two relations.

The whole characteristics can be divided into common attributes and features provided by tags. Common attributes are the following:

- *id* – identification number of node, way or relation - an integer greater than or equal to 1;
- *user*, *uid* – the display name and numeric user id of the user who last modified the object (element);
- *timestamp* – time of the last modification of element;
- *visible* – whether the object (element) is deleted or not in the database;
- *version* – the version of the object (element);
- *changeset* – the changeset in which the object (element) was created or updated;
- for node: *longitude* and *latitude* can be also counted as common attributes.

As it was mentioned above, all elements have their tags (node which is the onliest part of other elements can't be taken into account). A *tag* is not an element, but a property attached to a node, way or relation. These properties are Key-Value pairs which describe the element. Keys are unique, they don't occur twice for one element. For example, if element NODE has a tag with Key equal "highway" and, respectively, Value equal "traffic_signals", it means that this node on a map corresponds to a traffic light as part of transport infrastructure and it can't indicate components of such infrastructure (consequently, can't have any other tag with key "highway"). A common structure of tag idea is shown in the Figure 4.9.

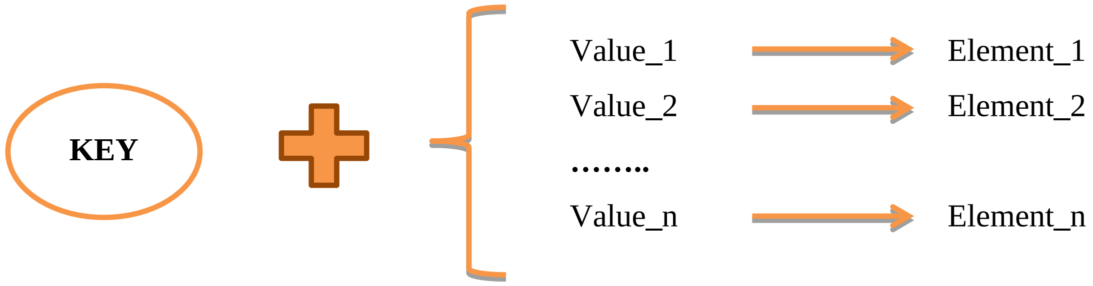


Figure 4.9 – A common structure of tag idea

Various tagging marks exist. A document called Map Features contains the full list of all certain tagging standard.

The tags are classified into several groups:

1. Physical (highway, building, barrier):

   For example, to describe roads we use the Highway. The tag consists of a Key that equals to "highway" and a Value that can have different values. It depends on which kind of road we bear in mind. If the road is described as a major highway which has divided carriageways, at least two lanes in both directions, emergency hard shoulder and the same characteristics as autobahn, its Value will equal to "motorway". If the road is intended for bicycle, its Value will equal to "cycleway". These tags describe such element as way.

2. Non-physical (route, sport, boundary):

   For example, to describe maritime, administrative or other boundary we use the Boundary. In this case of this example the Key equals to "boundary" and Value equals to "maritime" and "administrative". Boundary tag describes area.

3. Additional properties (addresses, annotation, name, restrictions, references, properties):

   a) Naming or a name tag is used to give individual name for an element. The Key equals to "name" that means the common default name. Also Key will equal to "loc_name", if it is a local name, etc. All Values are defined by the user. Name tag can be attached to a node, way or area.

   b) Addresses are used to give postal information and describe node or area. For example, to show the house number the Key should equal to "addr:housenumber" and the user defines the Value himself.

1.4. Download the OSM data

Every user can download map data from the OpenStreetMap. Data is downloaded in the form of XML formatted .osm files. For example, the file may be map.osm.xml. XML is Extensible Markup Language. This language has a number of rules which make possible to read a document in such format both for human and for machines. In principle, OSM XML is a list of data primitives which can include nodes, ways and relations. The Figure 4.10 shows examples of each data primitives' element in XML-file.

XML example of node.

```
<node id='251387902' timestamp='2008-09-10T18:12:29Z' uid='26974' user=‘master'
visible='true' version='2' changeset='595' lat='51.0303401' lon='13.7302577'>
<tag k='highway' v='traffic_signals' />
</node>
```

XML example of way.

```
<way id='34689233' timestamp='2009-11-19T11:18:06Z' uid='22542' user='Conny' visible='true'
version='5' changeset='3158816'>
<nd ref='403751787' />
<nd ref='567000301' />
<nd ref='554239613' />
<tag k='highway' v='footway' />
<tag k='lit' v='yes' />
</way>
```

XML example of relation.

```
<relation id='53928' timestamp='2010-04-27T08:45:45Z' uid='204738' user='viw' visible='true'
version='159'
changeset='4539138'>
<member type='relation' ref='548927' role='' />
<member type='relation' ref='659695' role='' />
<member type='relation' ref='611640' role='' />
<member type='relation' ref='550041' role='' />
<tag k='bus' v='urban' />
<tag k='by_demand' v='no' />
<tag k='by_night' v='yes' />
<tag k='line' v='bus' />
<tag k='name' v='61' />
<tag k='network' v='VVO' />
<tag k='operator' v='DVB' />
<tag k='service' v='busway' />
<tag k='type' v='line' />
</relation>
```

Figure 4.10 – Elements of each data primitives in the XML-files example

The Figure 4.11 presents a part of city which was exported from the OSM.

Figure 4.11 – Randomly chosen part of a city

In the Figure 4.12 the code is shown. This code is an abridged version of OSM XML example related to the Figure 4.11.

The structure of the OSM XML is the following:

- the first line shows the type of Markup Language used when creating this file (XML version 1.0) and type of encoding (UTF-8);
- the second line consists of information about the API version ("0.6"), generator which got this file, copyright, attribution and license;
- a block of nodes includes a number of such elements where every of them has its common attributes together with information about location. Some nodes have also a number of tags to define a concrete object represented by this element;
- a block of ways where each way has its common attributes, a list of nodes referenced to this way and minimum one tag to define the object;
- a block of relations has the same structure as block of ways with only one difference: instead of node list each relation has a list of its members.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2" copyright="OpenStreetMap and contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
 <bounds minlat="52.4604810" minlon="13.3842230" maxlat="52.4610300" maxlon="13.3853070"/>
 ...
<node id="442387117" lat="52.4605838" lon="13.3845350" user="wheelmap_visitor" uid="290680"
visible="true" version="4" changeset="8528990" timestamp="2011-06-24T13:25:45Z">
  <tag k="amenity" v="bank"/>
  <tag k="atm" v="yes"/>
  <tag k="name" v="Commerzbank AG"/>
  <tag k="wheelchair" v="yes"/>
 </node>
 ...
<way id="171372854" user="anbr" uid="43566" visible="true" version="2" changeset="14376604"
timestamp="2012-12-23T11:25:37Z">
  <nd ref="1824063565"/>
  <nd ref="274227352"/>
  <tag k="highway" v="primary"/>
  <tag k="lanes" v="2"/>
  <tag k="maxspeed" v="50"/>
  <tag k="name" v="Tempelhofer Damm"/>
  <tag k="oneway" v="yes"/>
  <tag k="postal_code" v="12099"/>
  <tag k="ref" v="B 96"/>
 </way>
 ...
<relation id="1105326" user="fx99" uid="130472" visible="true" version="7" changeset="13726733"
timestamp="2012-11-02T19:42:59Z">
  <member type="way" ref="70251559" role="outer"/>
  ...
  <member type="way" ref="188809964" role="outer"/>
  <tag k="boundary" v="postal_code"/>
  <tag k="note" v="12103 Berlin Tempelhof"/>
  <tag k="postal_code" v="12103"/>
  <tag k="type" v="boundary"/>
 </relation>
</osm>
```


Figure 4.12 – Code of OSM XM- file related to randomly chosen part of a city

Different ways of loading from OpenStreetMap exist. One file that consists of all the nodes, ways and relations of the OSM map is Planet.osm. Planet.osm contains a huge amount of entire planet data. The size of XML file is over 250GB uncompressed data or 16GB compressed. Planet.osm version is released every Thursday morning on weekly basis.

There are files that contain data for individual countries and areas. These files called Extracts and some of them can be downloaded in suitable file sizes (countries, parts of region etc.).

Country and area extracts are updated daily by producing regular differences. The size of daily diff is about 40 MB compressed.

To download small amounts of data you can specify the necessary area by a bounding box. It shows minimum and maximum latitude and longitude of chosen region (Figure 4.13). The smaller area allows downloading small data sets more quickly than in the case of larger region. Also it must be mentioned that the size of the box is restricted and can be bigger than ¼ degree in each dimension.

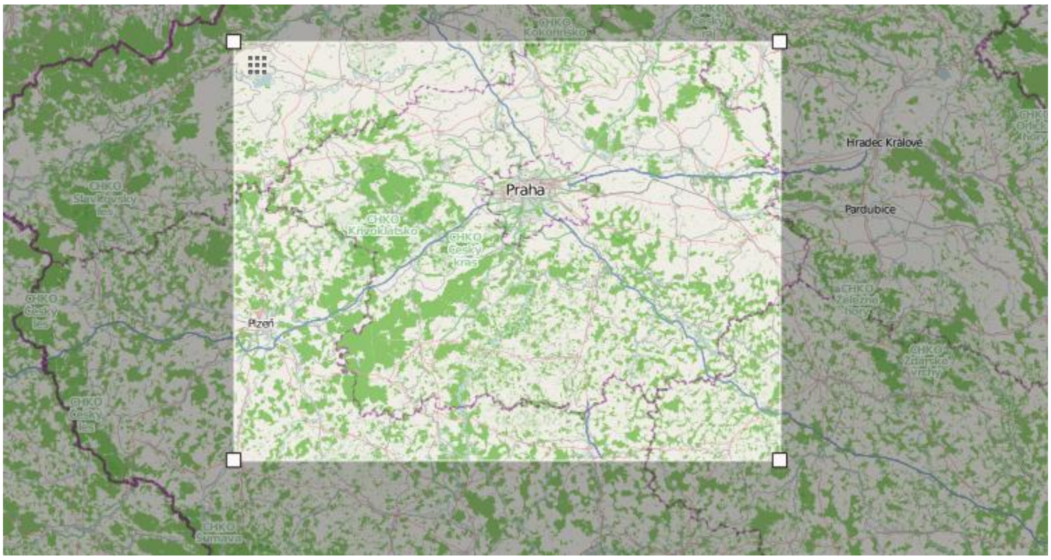


Figure 4.13 – Choice of a reference area to download it as bounding box

Data can be exported in several formats:

- OpenStreetMap XML Data that contains nodes, ways, relations and tags;
- Mapnik Image that supports export of PNG, JPEG, SVG, PDF;
- Embeddable HTML that creates HTNL code which can be used on any web page and always show latest data.

An API of OSM intends for getting and saving raw geodata to or from the OpenStreetMap database. Its main function is requesting the "map". All API calls orientate on manipulations with data composed of the basic elements: to read, create, delete or update this data. There is an example of API call which reads a bounding box with certain coordinates as XML-file:

http://api.openstreetmap.org/api/0.6/map?bbox=11.54,48.14,11.543,48.145

API has the following usage policies:

1. Requirements: license attribution must be displayed clearly and copyrights must be preserved.
2. Technical Usage Requirements: maximum two download threads, valid User-Agent identifying application and HTTP Referer.

OpenStreetMap has some download constraints related to the size of a file and total downloading. To get these restrictions of each API version the following API call must be created:

http://api.openstreetmap.org/api/0.6/capabilities

This call returns an XML-file (Figure 4.14).

```
<osm version="0.6" generator="OpenStreetMap server" copyright="OpenStreetMap and contributors" attribution="http://www.openstreetmap.org/copyright" license="http://opendatacommons.org/licenses/odbl/1-0/">
<api>
<version minimum="0.6" maximum="0.6"/>
<area maximum="0.25"/>
<tracepoints per_page="5000"/>
<waynodes maximum="2000"/>
<changesets maximum_elements="50000"/>
<timeout seconds="300"/>
<status database="online" api="online" gpx="online"/>
</api>
</osm>
```

Figure 4.14 – XML-file of API restrictions

This XML document is an example and returned values may change.

- In the line concerning minimum and maximum version the versions API accepted by server is shown.
- Value of area maximum presents the maximal size of area in square degrees which can be processed by API calls.
- Value "tracepoints per_page" equals to the maximum number of points included in a single GPS trace.
- Value "waypoints maximum" equals to the maximum number of nodes included in one way.
- Value "changesets maximum" equals to the maximum number of elements which can be a part of a changeset.

1.5. Map rendering

OpenStreetMap looks like an informative map that contains not only roads, railways, area boundaries and buildings, but also walking and cycling paths, banks, shops, stations, even the location of traffic lights and parking. When zooming the map it becomes more detailed. The Figure 4.15 is an example of a Standard OpenStreetMap.

Figure 4.15 – An example of Standard OpenStreetMap

OpenStreetMap allows to find the location of places according to their addresses or names of area, even according to the place name. For example, it is possible to fill in the search restaurant “Vapiano” or bank near “place name” and OSM shows all restaurants or banks locations.

Furthermore, OSM data has necessary information for routing for any transport modes: car, bicycle, foot. A lot of services and applications are based on the data, where OpenStreetMap is used for navigation.

The OSM has four types:

1. Standard Map looks like usual map with forests, rivers, roads and buildings.
2. Cycle Map emphasizes cycle routes and infrastructure. It also shows boundaries of green areas and reliefs.
3. Transport Map shows the railway and public transport routes and stations. There are also line numbers in the case of public transport.
4. MapQuest Open emphasizes the main roads and their numbers.

OpenStreetMap data is very rich and there are a lot of programmes that analyze the data at a certain level and for specific purposes. One of them is ITO Tools. It offers web-based services for transport users by collecting, managing, analyzing and presentation of transport data.

However, currently there is no open and free software, which can analyze OSM data for transport researches. But such types of researches and their tasks are so specific that it will be difficult to create universal programme to get various information from the OpenStreetMap.

1.6. Using the OSM data and license

Everyone who wishes to use OpenStreetMap data must specify "© OpenStreetMap contributors". The credit must be seen in the corner of the map, even if the map is browsable.

To show that a person uses data under the Open Database License and map images licensed as CC-BY-SA he/she can post link to copyright page (http://www.openstreetmap.org/copyright). It will be enough to put the name of license and link to it, if user distributes OSM in data form. In the case of printed works user must note the source as openstreetmap.org, opendatacommons.org, and if needed, creativecommons.org.

OpenStreetMap data is licensed under the Open Data Commons Open Database License (ODbL). The license is based on the share-alike principle. According to ODbL user can "*copy, distribute, transmit and adapt data as long as you credit OpenStreetMap and its contributors. If user alters or builds upon OSM data, he may distribute the result only under the same license*" (from http://www.openstreetmap.org/copyright/).

Documentation and cartography of OSM maps are licensed under the Creative Commons Attribution-ShareAlike 3.0 license (CC-BY-SA) that allow free sharing, remixing and making commercial use of the work. User must follow the rules of Attribution and Share Alike (the work must be attributed. Altered and transformed work may be distributed only under the same license).

2. Bicycle infrastructure data and its analysis

As it was written above, development of the automated analysis method for the assessment of bicycle infrastructure and, consequently, of software includes three stages. The first stage was presented in the previous subchapter "OpenStreetMap as data source" (a technical side of data obtaining will be described in the subchapter "OSM software"). Two other stages will be discussed in the actual one.

This part of theses has an aim to present and to explain the main ideas of the following issues:

- what kinds of bicycle infrastructure data are required for a new method and its software (stage 2);
- how collected data can be analyzed and assessed and what kinds of outputs can be obtained (stage 3).

2.1. Required infrastructure data

By returning to the subchapter 5 of the chapter 1 there are a number of questions which were established and must be answered for successful development of the automated analysis method for bicycle infrastructure assessment. First two of them were:

- What kinds of infrastructure data are required to make an assessment in the perspective of cycling mobility?
- Where and how this infrastructure data can be gathered?

If the second question was successfully solved by finding such data source as OpenStreetMap which includes all possible infrastructure data about the most objects in the EU, the first one was left without answer until now.

On the one hand, the extractions of concrete information from a huge database and its analysis and assessment will be, for sure, much easier task in comparison with evaluation and analysis of the whole amount of data. On the other hand, assessment results must be representative and meaningful. By these reasons it is actually to refer to the BICY project activities *indicators calculation* and *infrastructure analysis*.

The activity *indicators calculation* within the score of the BICY had an aim to calculate a number of indexes which will describe and assess a current situation with cycling mobility in the concrete place. Also these characteristics created an opportunity to compare different places between each other. The half of these indexes connects with the assessment of bicycle infrastructure and includes standard indicators and BICY index with its components.

Thus, a basis for required infrastructure data can be “infrastructure” indexes of *indicators calculation*. According to them, a minimal list of necessary information must consist of:

- length of cycle tracks (km) – all infrastructure facilities which were created only and specially for bicycle;
- length of roads (km) – all infrastructure facilities which are available for driving a motor vehicle;
- area (sq. km) – an area of a reference district or region;
- length of effective bike path (km):

  It includes not only exclusive bike lanes, but also traffic calming zones or streets with speeds less that 30km/h. Bidirectional lanes count only as one line. Pedestrian-only zones should not be counted.
- number of public bicycle parking;
- number of public car parks.

As it can be seen, listed insfrastructure data includes mostly common values: total length of roads, total length of cycle tracks, etc. But as it is known from practice, bicycle network facilities have a quite complex classification, and it will be interesting to consider this aspect much deeper. Unfortunately, almost all countries have their own classifications which are normally inconsistent between each other. That is why it is necessary to turn to *OSM data collection* and *infrastructure analysis*.

Based on these two activities, OSM classification of bicycle infrastructure was done (Figure 4.16).

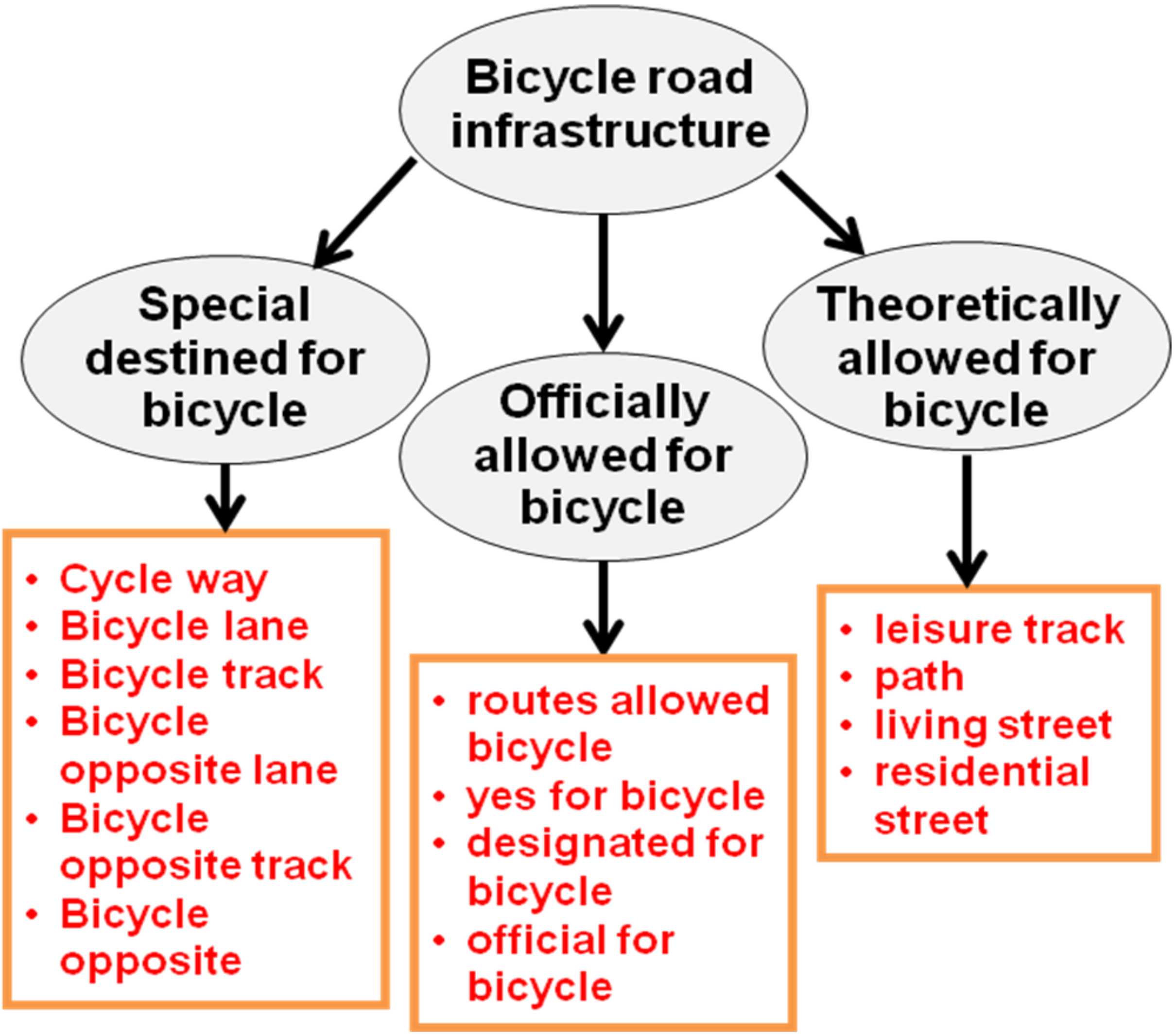


Figure 4.16 – Bicycle infrastructure classification accordning to OSM data

Map features of OSM were researched to make such kind of sorting process. According to these features, the main Key of OSM tags concerning road infrastructure is "highway". Its Values include and describe mostly all elements and objects of road networks for all modes of transport. In accord with the "highway" Key, bicycle road infrastructure can be divided into 3 groups: roads which were specially designed only for bicycle (left column in the Figure 4.16), roads which were officially available for cycling (middle column in the Figure 4.16) and roads which can be theoretically used for bicycle ride without any restrictions.

Such kinds of cycle facilities as bicycle parking, rental points and shops stand separately, but also can be counted.

As it was declared above, needs of *indicators calculation* include not only bicycle infrastructure data, but also a number of road network elements for other modes of transport. This is one of the reasons why OSM data of common road infrastructure must be also obtained. They consist of:

- motor way;
- motor way link;
- trunk;
- trunk link;
- primary road;
- link of primary road;
- secondary road;
- link of secondary road;
- tertiary road;
- link of tertiary road;
- residential road;
- unclassified road;
- road;
- living street;
- service road;
- track;
- foot way;
- path;
- steps;
- platform;
- pedestrian zone.

As it is seen, some listed road types are a part of the bicycle infrastructure elements with which they were included after *infrastructure analysis*.

As it was mentioned above, OSM data is quite reach and can give often a number of additional information concerning different infrastructure objects. One of the most interesting parameters is surface types of roads, both for bicycles and cars. Classification of this parameter is shown in the Figure 4.17.

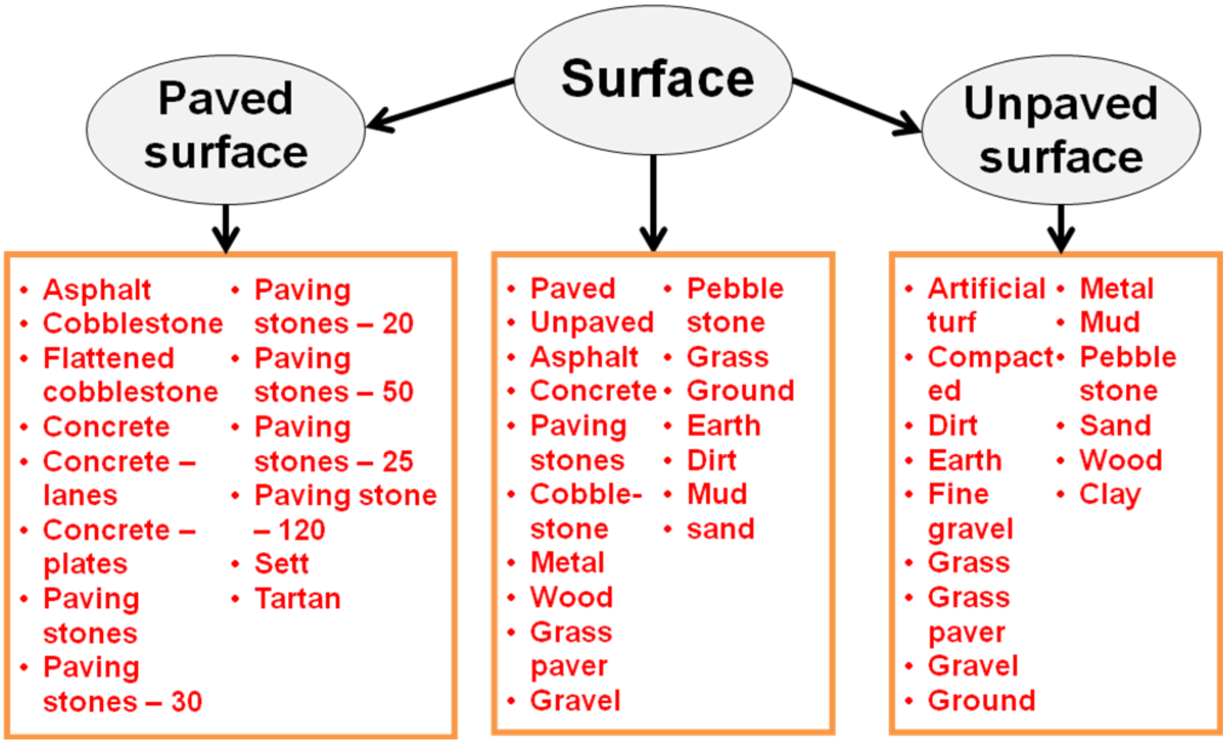


Figure 4.17 – Classification of surface types according of OSM data

The Key attribute "surface" can take itself a number of values (middle column in the Figure 4.17). In its presence for the same element, parameters "paved" and "unpaved" can be Key attributes (left and right columns in the Figure 4.17).

This whole detailed classification of all road network components can help to get a full picture not only about bicycle facilities, installations and road elements, but also about the same elements of competitive mode of transport, especially, cars. According the described data, calculation of all cycling indicators can be done. All these actions will help to carry out analysis and assessment of the bicycle infrastructure.

2.2. Approaches for the assessment of bicycle infrastructure data

Two possible approaches were used as parts of considered automated analysis method. They will be discussed below.

As it was mentioned above, the first analysis approach for the assessment of bicycle infrastructure follows from the activity *indicators calculation*. A special research was made in this direction by UNIBO-DICAM. It explains the relevance of these indicators for assessment of bicycle infrastructure and its efficiency. High dependence of bicycle mode share on cycling index is one of the most representative diagrams which shows benefits and usefulness of this approach (Figure 4.18).

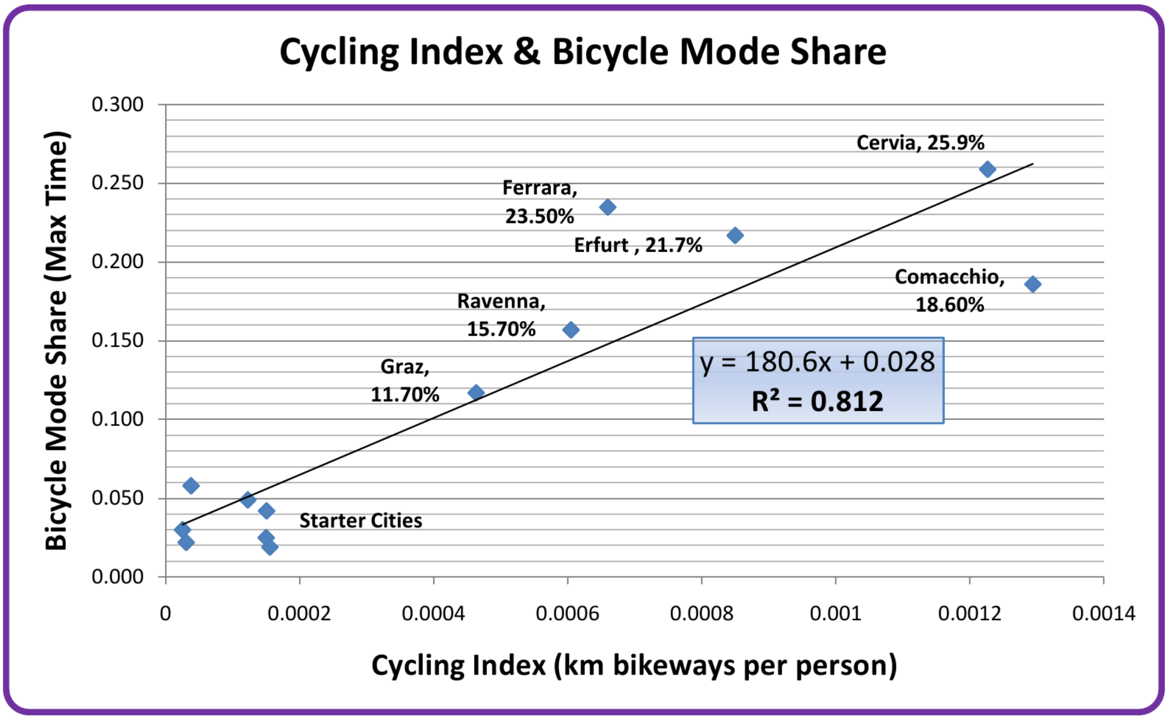


Figure 4.18 – Correlation between cycling index and bicycle mode share

Detailed information about above-mentioned research of correlation between cycling indicators and bicycle usage can be found in the Common Indicators Report [13].

Quantitative analysis of the transport infrastructure data including bicycle network elements is the second analysis approach for the assessment of bicycle infrastructure. This approach originated from analysis of available OSM infrastructure data which was done above. According to it, a table of necessary information was created. Example of such tables can be found in Annex A.

As it was presented, total amount of necessary infrastructure data is quite large. To process and to analyze it, to make any assessment, special OSM software was created which will be described below.

3. OSM software

All theoretical stages of the automated analysis method development were considered in previous subchapters. As it was also written above, OSM software is a realization form of this method for the assessment of bicycle infrastructure. It means that implementation of presented ideas will be described and explained in this subchapters in general.

In the perspective of this OSM package and its development, a number of questions will be also discussed below. Among them are software structure, its functions and outputs, development environment including programming language and its modules which were used by creating a package.

3.1. Development Environment

The software was developed in an open source environment Linux. A basis for the future package was Python Programming Language which has two following advantages:

- all versions of this language is free to download and to use them;
- there is a number of sophisticated extensible and customizable modules which include different functions and opportunities and, correspondingly, can simplify a creation work.

Python version 2.7 was chosen as one of the latest releases by the time of development start.

There is also a list of Python modules which are not a part of standard package installed automatically with programming language. They were mostly selected in accordance with goals of the future software:

- *SciPy* and *NumPy* are two important Python applications which give us opportunity to use various special mathematical functions;
- *Pickle module* converts Python objects into a stream of bytes and vice-versa.
- *NetworkX* is a module which allows to work with different graphs (allows the generation of visual maps such as the one pictured below)
- WXPython and Matplotlib: these two applications give a possility to generate visual maps and different pictures based on graphs.

3.2. Software structure

Module *OpenStreetLib* was the main component of OSM software. It looks like a number of functions which combinations allow to make decisions of above-assigned task. In common, all components of module can be divided into three spheres (Figure 4.19).

More in detail, all functions can be divided into 6 group united by common aim:

- Group 1 – Functions for writing/ reading information getting from OpenStreetMap;
- Group 2 – Functions for processing XML-files: parsing, getting and saving OSM data;
- Group 3 – Functions for processing of OSM data;
- Group 4 – Functions for getting graphs with different configuration;
- Group 5 – Functions for drawing graphs in different ways;
- Group 6 – Functions for parameters calculation.

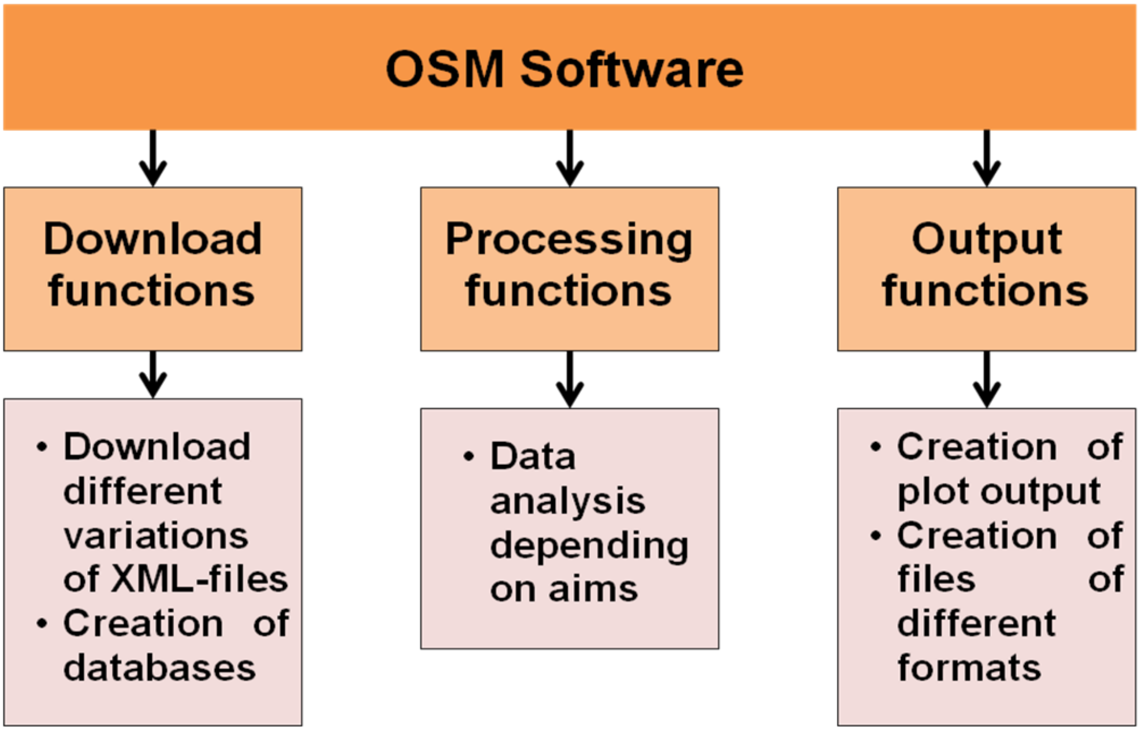


Figure 4.19 – Common structure of OpenStreetLib module

3.3. OpenStreetLib Functions

As it was mentioned above, modul OpenStreetLib includes a number of functions used for obtaining, processing and analysing a data. These functions will discussed here only in general terms: function name and sense, also its inputs and outputs. The full version of programming code can be found in Annex B.

Group 1 – Functions for writing/ reading Information getting from OpenStreetMap:

- ***save_graf_as_txt*** *(graf, filename=None, attr = None, value=None)*:

  Sense: information about the created graf is saved in a txt-file.

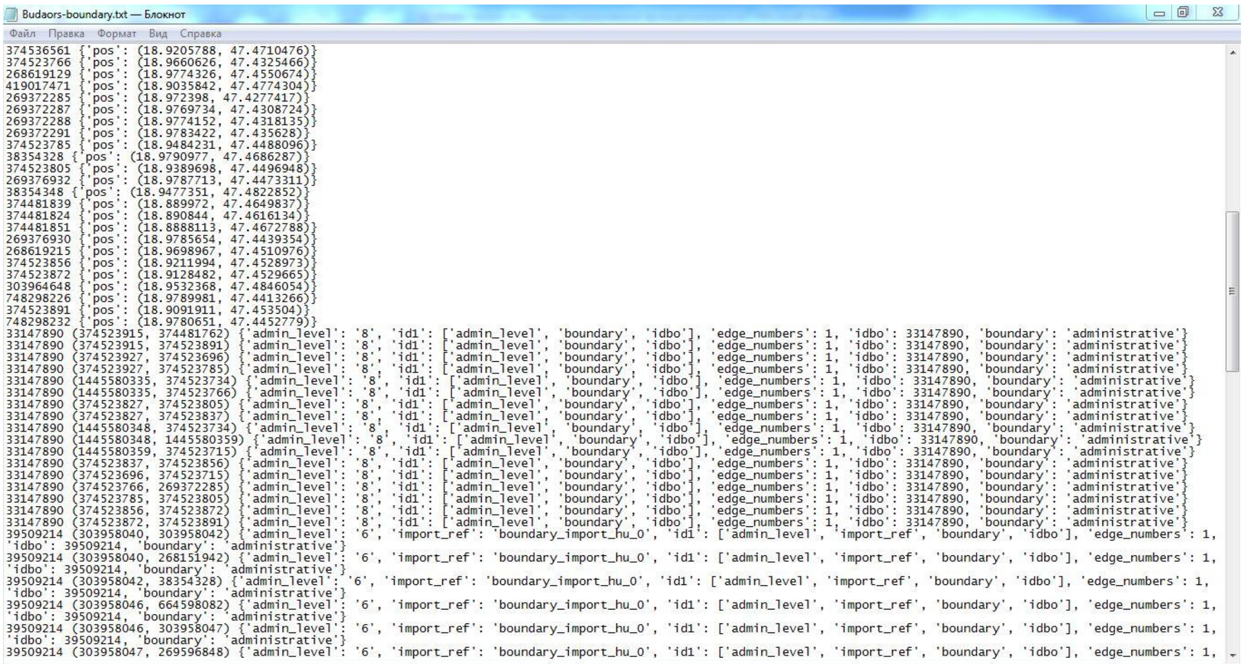
  Inputs:

  "graf" is a graph which will be a basis to create a new graph according to filter parameter.

  "filename" is a name of future created txt-file. It is not obligatory parameter and, if it is not given, file name will be "graf-boundary.txt".

  "attr" and *"value"* are parameters to filter a new graph. They correspond to Key and Value of OSM tag, respectively. For example, default value of *"attr"* parameter equals "boundary" and, consequently, a new graph will be a graph which includes only elements of different boundary types (key part of OSM tag).

  Outputs: new graph according to values of *"attr"* and *"value"* parameters and txt-file with information about nodes and edges (Figure 4.20).



```
Budaors-boundary.txt — Блокнот
Файл Правка Формат Вид Справка
374536561 {'pos': (18.9205788, 47.4710476)}
374523766 {'pos': (18.9660626, 47.4325466)}
268619129 {'pos': (18.9774326, 47.4550674)}
419017471 {'pos': (18.9035842, 47.4774304)}
269372285 {'pos': (18.972398, 47.4277417)}
269372287 {'pos': (18.9769734, 47.4308724)}
269372288 {'pos': (18.9774152, 47.4318135)}
269372291 {'pos': (18.9783422, 47.435628)}
374523785 {'pos': (18.9484231, 47.4488096)}
38354328 {'pos': (18.9790977, 47.4686287)}
374523805 {'pos': (18.9389698, 47.4496948)}
269376932 {'pos': (18.9787713, 47.4473311)}
38354348 {'pos': (18.9477351, 47.4822852)}
374481839 {'pos': (18.889972, 47.4649837)}
374481824 {'pos': (18.890844, 47.4616134)}
374481851 {'pos': (18.8888113, 47.4672788)}
269376930 {'pos': (18.9785654, 47.4439354)}
268619215 {'pos': (18.9698967, 47.4510976)}
374523856 {'pos': (18.9211994, 47.4528973)}
374523872 {'pos': (18.9128482, 47.4529665)}
303964648 {'pos': (18.9532368, 47.4846054)}
748298226 {'pos': (18.9789981, 47.4413266)}
374523891 {'pos': (18.9091911, 47.453504)}
748298232 {'pos': (18.9780651, 47.4452779)}
33147890 (374523915, 374481762) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523915, 374523891) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523927, 374523696) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523927, 374523785) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (1445580335, 374523734) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (1445580335, 374523766) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523827, 374523805) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523827, 374523837) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (1445580348, 374523734) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (1445580348, 1445580359) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (1445580359, 374523715) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523837, 374523856) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523696, 374523715) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523766, 269372285) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523785, 374523805) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523856, 374523872) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
33147890 (374523872, 374523891) {'admin_level': '8', 'id1': ['admin_level', 'boundary', 'idbo'], 'edge_numbers': 1, 'idbo': 33147890, 'boundary': 'administrative'}
39509214 (303958040, 303958042) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
'idbo': 39509214, 'boundary': 'administrative'}
39509214 (303958040, 268151942) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
'idbo': 39509214, 'boundary': 'administrative'}
39509214 (303958042, 38354328) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
'idbo': 39509214, 'boundary': 'administrative'}
39509214 (303958046, 664598082) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
'idbo': 39509214, 'boundary': 'administrative'}
39509214 (303958046, 303958047) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
'idbo': 39509214, 'boundary': 'administrative'}
39509214 (303958047, 269596848) {'admin_level': '6', 'import_ref': 'boundary_import_hu_0', 'id1': ['admin_level', 'import_ref', 'boundary', 'idbo'], 'edge_numbers': 1,
```


Figure 4.20 – Fragment of the txt-file

- ***save_graf_as_pickle*** *(graf, filename=None)*:

  Sense:

  A certain graph is saved as pickle-file which allows storing a graph data in a computer. The main advantages of such file are the following:

  - it takes less place in computer memory than othe file formats;
  - possibility of fast access to a data.

  Inputs:

  "*graf*" is a graph which must be saved.

  "*filename*" is a name of future pickle-file. A default value of this parameter is "graf.dat".

  Output: pickle-file.

- ***get_graf_from_pickle*** *(filename)*:

  Sense: this function makes an inverse operation respect to the previous function – to get a saved graph from pickle-file.

  Input: "*filename*" is a name of the pickle-file with data.

  Output: graph.

Group 2 – Functions for processing XML-files: parsing, getting and saving OSM data:

- ***get_tiles_from_square*** *(coords)*:

  Sense:

  In common, the function divides a chosen piece of OSM map into tiles and returns a list of their coordinates.

In detail, this function gets the coordinates of an initial bounding box (parameter "*coords*") and cuts it in pieces – small bounding boxes which size allows downloading them without any constrains. It looks like matrix which has a number of rows ("*nwidth*") and a number of columns ("*nlength*"). Every cell of this "matrix" has its own coordinates of left bottom and right top corners. All coordinates of these small map pieces are written in one list "*tiles*".

According to restrictions for the dimensions of small bounding box, its length takes a value of 0.0315 degree and its width – 0.0152 degree. These sizes were determined after a number of tests: attention was paid on maximal size of one box and maximal amount of node elements in this box.

Input:

"*coords*" is coordinates of a bounding box which was chosen for infrasturucture data collection (term "bounding box" was defined above in the subchapter "OpenStreetMap as data source"). This parameter is a constant list and includes longitudes and latitudes of two points: left bottom corner and right top corner of the selected bounding box.

Outputs:

"*tiles*" is a list of coordinates for small bounding boxes. Each element of "*tiles*" is a separate constant list by analogy with "*coords*".

"*nwidth*" and "*nlength*" are a number of rows and columns in the "matrix", respectively.

- ***get_graf_from_tile*** *(tile, graf=None)*:

Sense:

The function reads a piece of OSM map from website (openstreetmap.org) according to tile coordinates and saves it as XML-file under the name "data-city-temp_x.xml". Created XML-file is parsed with the help of class *Processing*. A result of parsing is an amount of nodes and edges with all available information which is obtained from their OSM tages. A graph composed of this data is generated in the end.

It must be mentioned that node corresponds to OSM element NODE and edge is a section of OSM element WAY. Every WAY is divided into edges and returns the following information to each of them:

  - two nodes with their coordinates;
  - data according to OSM tages of this WAY.

Inputs:

"*tile*" is a constant list. Its value is coordinates of bounding box or small bounding box (see previous function).

If it is necessary to upgrade an existing graph, a value of this graph must be returned to parameter "*graf*". Otherwise, a new empty graph will be generated.

Output:

"*graf*" is a graph which includes nodes and edges data of a chosen OSM map piece.

- ***get_xml_files*** *(tiles, nlength, nwidth, cityname)* :

  Sense:

  The function reads pieces of OSM map from website (openstreetmap.org) according to coordinates of "*tiles*" list and saves them as a number of XML-files (Figure 4.21).

  Inputs:

  "*tiles*" is a list of coordinates for small bounding boxes.

  "*nwidth*" and "*nlength*" are a number of rows and columns in the "matrix" of intial bounding box, respectively.

  "*cityname*" is a name of city which data must be download from OSM. It is used to generate names of XML-files.

  Output:

  Function downloads a number of XML files. Each of these XML-files is one small bounding box.


```
<?xml version="1.0" encoding="UTF-8"?>
- <osm generator="CGImap 0.0.2" version="0.6">
    <bounds maxlon="18.9559000" maxlat="47.4413000" minlon="18.9243000" minlat="47.4260000"/>
    <node version="7" timestamp="2007-12-10T13:58:16Z" changeset="232055" visible="true" uid="20650" user="busaipet" lon="18.9446905" lat="47.4204945"
      id="145508119"/>
    <node version="4" timestamp="2009-05-25T10:36:42Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9236612" lat="47.4225421"
      id="146473554"/>
    <node version="4" timestamp="2009-05-25T10:36:42Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9243049" lat="47.4232505"
      id="146474228"/>
    <node version="4" timestamp="2009-05-25T10:36:42Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9249401" lat="47.4240113"
      id="146475032"/>
    <node version="13" timestamp="2009-05-25T10:36:42Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9313290" lat="47.4243980"
      id="146477340"/>
    <node version="4" timestamp="2009-05-25T10:36:43Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9304964" lat="47.4252265"
      id="146484322"/>
    <node version="4" timestamp="2009-05-25T10:36:43Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9296264" lat="47.4249985"
      id="146484326"/>
    <node version="4" timestamp="2009-05-25T10:36:43Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9278365" lat="47.4242071"
      id="146484328"/>
    <node version="4" timestamp="2009-05-25T10:36:43Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9272984" lat="47.4242651"
      id="146484329"/>
    <node version="4" timestamp="2009-05-25T10:36:43Z" changeset="1314679" visible="true" uid="55843" user="flaktack" lon="18.9262362" lat="47.4246243"
      id="146484330"/>
    <node version="3" timestamp="2009-03-30T19:33:58Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9312658" lat="47.4245920"
      id="146487006"/>
    <node version="3" timestamp="2009-03-30T19:33:58Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9312658" lat="47.4248242"
      id="146487008"/>
    <node version="3" timestamp="2009-03-30T19:33:58Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9314203" lat="47.4254224"
      id="146487009"/>
    <node version="3" timestamp="2009-03-30T19:33:59Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9314460" lat="47.4257475"
      id="146487010"/>
    <node version="3" timestamp="2009-03-30T19:33:59Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9313516" lat="47.4260088"
      id="146487012"/>
    <node version="4" timestamp="2010-12-03T18:24:10Z" changeset="6529384" visible="true" uid="55843" user="flaktack" lon="18.9308825" lat="47.4268927"
      id="146487014"/>
    <node version="4" timestamp="2010-03-20T09:02:34Z" changeset="4177340" visible="true" uid="55843" user="flaktack" lon="18.9296821" lat="47.4290105"
      id="146487016"/>
    <node version="3" timestamp="2009-03-30T19:33:59Z" changeset="870348" visible="true" uid="36129" user="City-busz" lon="18.9292488" lat="47.4298064"
```


Figure 4.21 – Example of a XML-file

- ***get_graf_from_xml_direct*** *(filename, graf=None)*:

  Sense:

  This function is quite close to the function "***get_graf_from_tile*** *(tile, graf=None)*". Difference between them is the following:

"***get_graf_from_xml_direct***" reads data from OSM XML-file which has already been downloaded. "***get_graf_from_tile***" gets a XML-file from OSM serves in the beginning and only then processes it.

Inputs:

"*filename*" is a name of XML-file for parsing and data collection.

If it is necessary to upgrade an existing graph, a value of this graph must be returned to parameter "*graf*". Otherwise, a new empty graph will be generated.

Output:

"*graf*" is a graph which includes nodes and edges data from a given XML-file.

- ***get_graf*** *(coords, graf=None, cityname=None)*:

  Sense:

  This function includes a major part of all above-described functions and, thereby, gathers all their opportunities. In common, it gets coordinates of an intial bounding box and returns a graf with OSM data of nodes and edges.

  Inputs:

  "*coord*" is a constant list of coordinates.

  If it is necessary to upgrade an existing graph, a value of this graph must be returned to parameter "*graf*". Otherwise, a new empty graph will be generated.

  "*cityname*" is a name of town or region. According to the software idea, this name must be a name part of the relevant XML-file.

  Output: a new graph which includes all OSM data of the given bounding box.

Group 3 – functions for processing of OSM data:

- ***get_len*** *(graf, edge)*:

  Sense: a length of given edge will be calculated.

  Inputs:

  ID number of a certain edge returns as parameter "*edge*".

  "*graf*" is a graph where this edge is situated.

  Output: "*dist*" is a real distance in meters between two nodes of given edge.

- ***get_len_two_node*** *(graf, node1, node2, weight=None)*:

  Sense: the shortest lenght (in metes) between two nodes in graf will be calculated.

  Inputs:

  "*node1*" and "*node2*" are ID numbers of two nodes (their IDs),

  "*graf*" is a graph where these nodes are situated.

It it is necessary to take into account weights of graph edges, parameter "*weight*" must get a name of attribute which values will correspond to edge weights.

Output: "*lenght*" is the shortest lenght in meters between two nodes in graph

- ***is_intersected*** *(x11, y11, x12, y12, x21, y21, x22, y22)*:

  Sense:

  The function determines intersection of two lines: whether they cross each other or no.

  Inputs:

  There are two lines. Each line is represented by two points with X- and Y-coordinates. The first line has "*x11*", "*y11*", "*x12*", "*y12*", and the second one – "*x21*", "*y21*", "*x22*", "*y22*". Line parameters can also be given in vector format where each row corresponds to a line.

  Output:

  The result is a bool variable: it equals "true", if both lines intersect.

- ***r_return*** *(x,x1,x2)*:

  This function is a help-tool for the previous function "**is_intersected**".

- ***supernode_coord*** *(nodelist, IDjunction)*:

  Sense:

  The function calculates coordinats of *super node* by using a special method of location planning. "*Super node*" means a new center located node. This function relates to the perspective of the future software development.

  Inputs:

  "*IDjunction*" is an ID of researched intersection.

  "*nodelist*" is a list of nodes information. This information includes their IDs, coordinates and weights.

  Outputs: coordinates of a new node (super node) and its new ID

- ***new_coords*** *(graf, coords)*:

  Sense: this function converts geographic coordinate system to Cartesian coordinate system (Figure 4.22).

  Inputs:

  "*graf*" is a graph which will converted.

  "*coords*" is a bounding box coordinates of this graph.

  Outputs:

  "*newgraf*" is a new converted graph.

  "*newcoords_draw*" is a converted coordinates of bounding box.

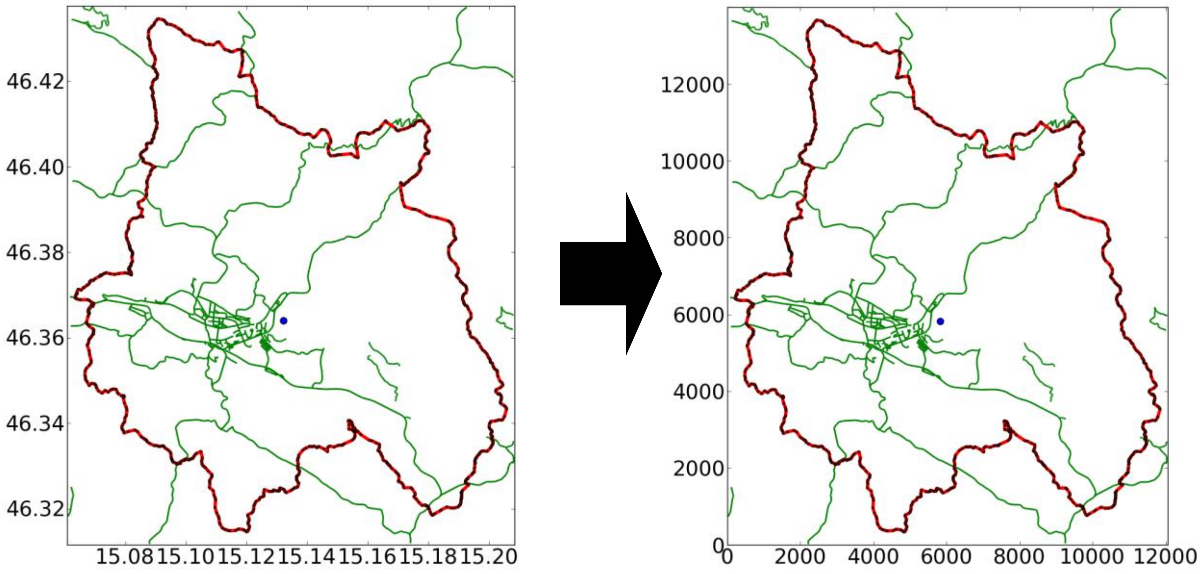


Figure 4.22 – Example of coordinates conversion

Group 4 – Functions for getting graphs with different configuration

- ***unite_two_grafs** (graf1, graf2)*:

  Sense: this finction unites together two different graph. It avoids also a double writing of the same nodes and edges to a new graph.

  Inputs:

  "*graf1*" and "*graf2*" are two graphs which must be united.

  Output: "*graf*" is a new united graph.

- ***square_boundary** (graf, coords)*:

  Sense: the function deletes all nodes and edges which are situated outside of the square boundary (with other words: outside of bounding box).

  Inputs:

  "*graf*" is an initial graph.

  "*coord*" is bounding box coordinates of this graph.

  Output: "*graf*" is a new graph without elements outside the boundaries.

- ***get_graf_one_key** (graf, attr)*:

  Sense:

  This function creates a graph which all edges have the same Key attribute getting from OSM tags. For example, all edges of graph have one of all Key attributes – "highway".

  Inputs:

"*graf*" is an intial graph.

"*attr*" is an attribute which corresponds to Key of tages.

Outputs: "*KeyGraf*" is a new graph which includes only edges with the same Key attribute.

- ***get_graf_one_key_clean*** *(graf, attr)*:

  This function does the same work as function "***get_graf_one_key***". In comparison with the previous one, there is only one difference: it cleans a new graph from the repeated information.

- ***get_graf_one_value*** *(graf, key, value)*:

  Sense:

  This function creates a graph. All their edges have the same tag pair of Key and Value. For example, all edges of graph have the same Key-Value pair – "highway"-"motorway".

  Inputs:

  "*graf*" is an initial graph.

  "*key*" and "value" correspond to a tag pair of Key and Value.

  Outputs: "*ValueGraf*" is a new graph which includes only edges with the same Key-Value attribute pairs.

- ***get_graf_one_value_clean*** *(graf, key, value)*:

  This function does the same work as the function "***get_graf_one_value***". The difference consists in the following: "***get_graf_one_value_clean***" is based on the function "***get_graf_one_key_clean***" and "***get_graf_one_value***" – on "***get_graf_one_key***".

- ***create_road_graf*** *(graf)*:

  Sense:

  This function allows getting a graf of roads. This graf includes only edges which have a Key attribute equal "highway". All possible values of this attribute are collected in the list *ROADATTRS*. There are "motorway", "motorway_link", "trunk", "trunk_link", "primary", "primary_link", "secondary", "secondary_link", "tertiary", "tertiary_link", "residential", "unclassified", "road", "living_street", "service"' "track". An example of such graph is presented in the Figure 4.19 with green color.

  Input: "graf" is an initial graph.

  Output: "RoadsGraf" is as new graph which includes only elements of road infrastructure.

- ***admin_boundary_graf*** *(graf, level=None, cityname=None)*:

  Sense:

The function allows creating a graph of administrative boundaries for city, region, etc. An example of such graph is presented in the Figure 4.23 with red color.

Inputs:

"*graf*" is an initiatl graph.

"*level*" is a level of boundary if such one exists.

"*cityname*" is a name of a certain city or region if such one is mentioned in the obtained OSM information.

Output: "*AdminGraf*" is a new graph which includes only elements of administrative boundaries.

Figure 4.23 – Example of graph processing for city Budaors: from initial one (on the left) to graphs of roads (green on the right) and administrative boundaries (red on the right)

- ***road_graf_in_boundary*** *(RoadsGraf, AdminGraf, GravitGraf, coords)* :

  Sense:

  This function creats a road graph which is covered only administrative territory (Figure 4.24).

  Inputs:

  "*RoadsGraf*" is a road graph of bounding box.

  "*AdminGraf*" is an administrative graph.

  "*GravitGraf*" is an auxiliary graph of gravitation points.

  "*coords*" is bounding box coordinates of a bounding box for intial graph.

  All mentioned graphs are generated normally from the same initial graph.

  Output: "*NewRG*" is a road graph which data covers only administrative territory of a certain city or region.

Figure 4.24 – Example of getting a road graph in which is covered only administrative territory (colors: red – administrative boundary graph, blue – auxiliary graph, green – road graph)

- ***is_road*** *(attrs)*:

  This function returns “true” if attributes of the list “*attrs*” correspond to roads.

- ***is_mainfeatures*** *(attrs)*:

  This function returns “true” if attributes of the list “*attrs*” correspond to a main features of OSM. These features (*MAINFEATURES*) include “aerialway”, “aeroway”, “amenity”, “boundary”, “craft”, “geological”, “highway”, “historic”, “leisure”, “man_made”, “military”, “natural”, “office”, “place”, “power”, “railway”, “route”, “shop”, “sport”, “tourism”, “waterway”, “barrier”, “building”, “emergency”, “landuse”, “public_transport”.

- ***has_attr*** *(attrs,attrs_comp)*:

  This function returns “true” if at least one attribute in a list “*attrs*” is in a list “*attrs_comp*”. For example, it will be “true” in such situation: “*attrs*” includes “*author*” and “*highway*“, “*attrs_comp*” consists of “*highway*” and “*road*”.

- ***get_graf_weighted_nodes*** *(graf)*:

  Sense:

  This function creates a graph of all nodes (only nodes, without edges) which are weighted depending on a number of edges connected with each node. With other words, each node in the graph will get an additional attribute “weight” which can be equaled 1 (if node belongs only to one edge), 2 (if node belongs to two edge) and so on. Such kind of attribure will help to find, for example, intersection on the road network: by processing of a road graph, all nodes with “weight” equal 3 or more will be junctions.

  Input: “*graf*” is a graph under study. It can be, for instance, a road graph.

  Output: “*WeightNodes*” is a new graph of weighted nodes. The Figure 4.25 shows a result of plotting such graph together with a road graph.

- ***get_graf_with_edges_lenghts*** *(graf)*:

  Sense:

  This function creates a graph of nodes and edges. All it edges are weighted with length: each edge has a new attribute "length" which equals length of this edge in meters.

  Take on graph and returns the same one with weighted edges

  Input: “*graf*” is a graph under study.

  Output: “*WeightEdges*” is a graph which has only one difference with “*graf*”: an additional attribute "length" for each edge.

- ***get_full_graf_weighted_nodes*** *(graf)*:

This function creats a full graf of nodes and edges which nodes are weighted with the help of ”***get_graf_weighted_nodes***“.

Input: “*graf*” is a graph under study.

Output: “*WeightGraf*” is a graph which has only one difference with “*graf*”: their nodes are weighted. Example of such graph is shown in the Figure 4.25 (here: “*graf*” is a road graph). It looks the same as an image of “*WeightNodes*” together with a road graph.

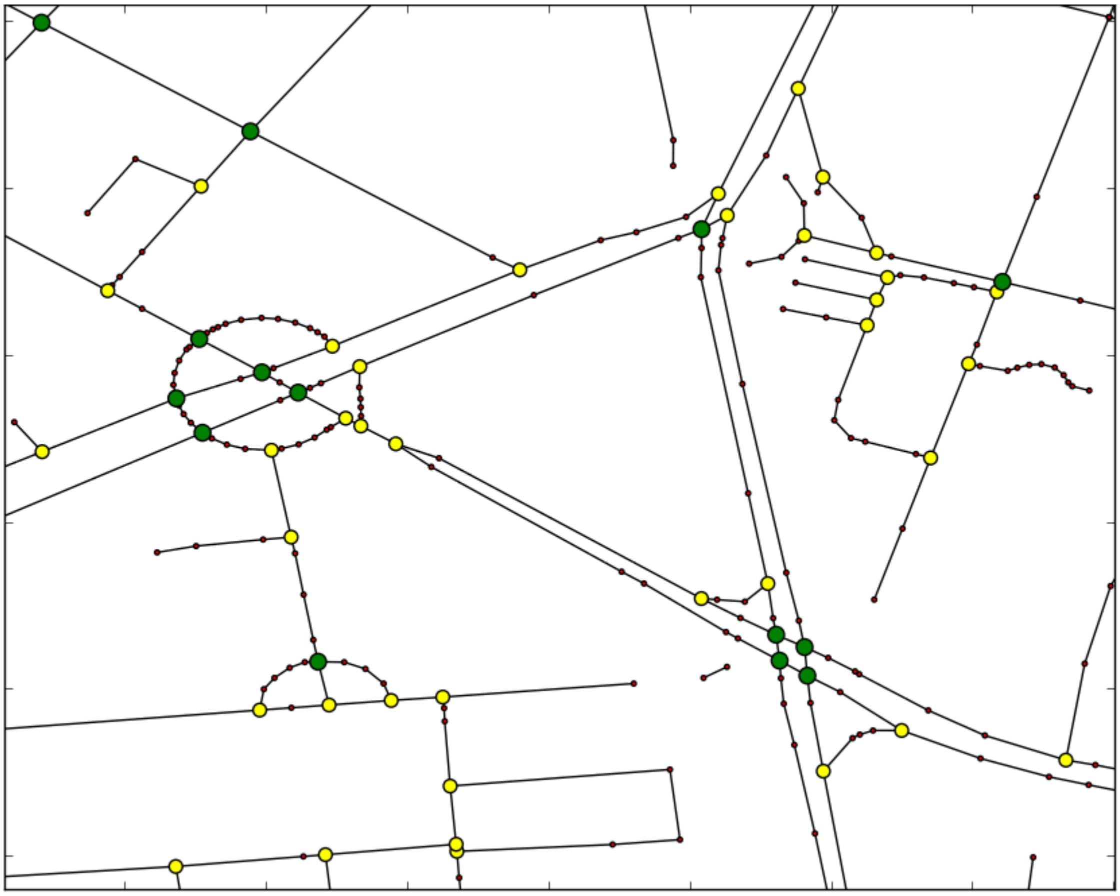


Figure 4.25 – Example of graph with weighted nodes (yellow nodes – weight equals 3, green node – weight equals 4)

Group 5 – Functions for drawing grafs in different ways:

- ***draw_graf*** *(graf, coords=None)*:

  Sense: this function creates an image of a given graph.

  Input:

  “*graf*” is a given graph which must be drawn.

  “*coord*” is bounding box coordinates of this graph.

  Output: an image of a given graph.

  Figure 4.26 presents an image example of an intial graph (a raw graph which includes all types of nodes and edges).

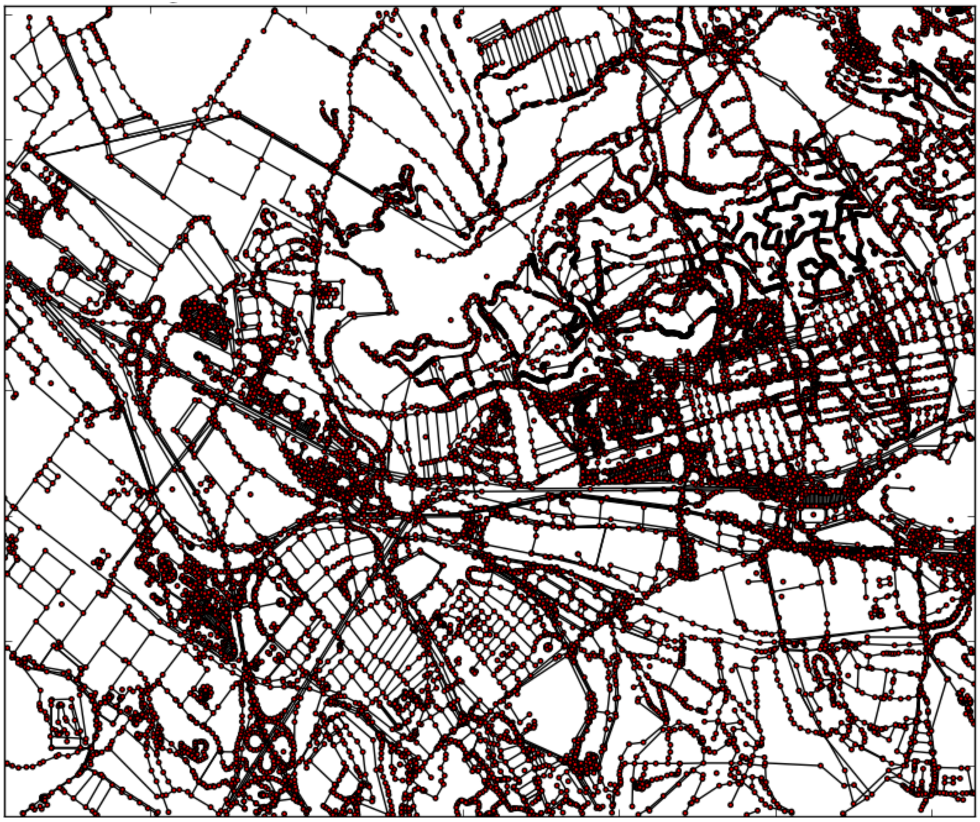


Figure 4.26 – An image example of an intial graph

- ***draw_weighted_nodes** (graf, WeightNodes, coords=None)*:

  Sense:

  This function creates a graph picture where the weight of each node is showed. It was shown in the Figure 4.25.

  Inputs:

  "*graf*" is a given graph which must be drawn.

  "*coord*" is bounding box coordinates of this graph.

  "*WeightNodes*" is a graph of weighted nodes based on "*graf*".

  Outputs: an image of graph with weighted nodes (the Figure 4.21).

- ***draw_grafs_as_layers** (RoadsGraf, AdminGraf, SupernodeGraf=None, G=None, T=None, H=None, coords=None)*:

  Sense:

  The function draws different graphs as layers: one graph - one layer.

  Inputs:

  "*coord*" is bounding box coordinates of an initial graph.

  "*RoadsGraf*" is a road graph.

  "*AdminGraf*" is an administrative graph.

  "*SupernodeGraf*" is a graph of super nodes (this graph type will be discussed below).

"*G*", "*T*" and "*H*" are graphs which are based on the initial one.

Output: a color image where each graph layer has its own color.

- ***draw_road_types*** *(AdminGraf, MotorwayFull=None, TrunkFull=None, PrimaryFull=None, SecondaryFull=None, TertiaryFull=None, ResidGraf=None, LivingStrGraf=None, RoadGraf=None, TrackGraf=None, ServiceGraf=None, UnclassGraf=None, CyclewayGraf=None, CyclelementsGraf=None, BiciGraf=None, coords=None)*:

  Sense:

  The function draws different graphs of road types as layers: one graph - one road type - one layer.

  Input:

  "*coord*" is bounding box coordinates of this graph.

  Each graph of list corresponds to one types of road according OSM.

  Output: a color image where each road type has its own color (Figure 4.27).

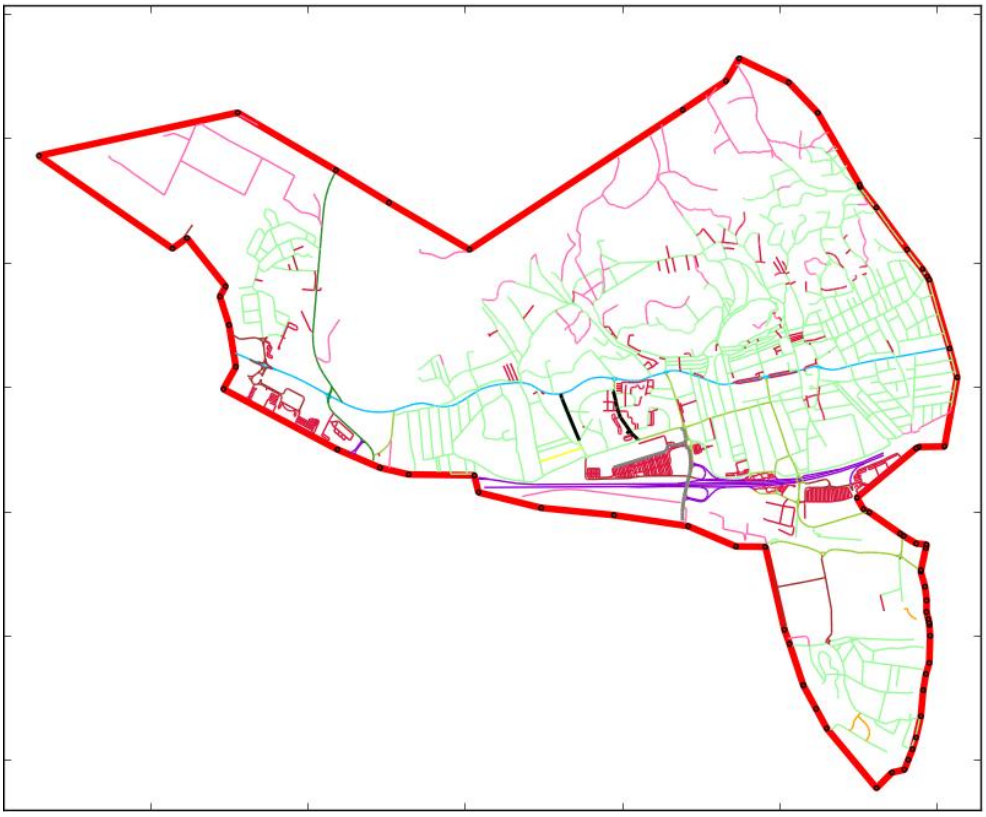


Figure 4.27 – An example of color image of graph layers: red color – administrative boundary graph, blue color – auxiliary graph, green color – "RoadsGraf"

Group 6 – Functions for parameters calculation:

- ***get_length_key_value*** *(graf, key, value, mainkey=None, mainvalue=None)*:

Sense:

The function calculates a total road length. Type of road is set by Key and Value. Parameters "*mainkey*" and "*mainvalue*" make possible to use two levels of Key-Value filter.

Inputs:

"*graf*" is a graph under study.

"*key*", "*value*", "*mainkey*", "*mainvalue*" are tags pairs of different levels.

Output: a total road length of a certain type.

- ***return_list_of_keys*** *(graf, key)*:

  Sense:

  This function returns a list of Values corresponding to a given Key.

  Inputs:

  "*graf*" is a graph under study.

  "*key*" is a Key attribute of tags which Values must be extracted.

  Output: a list of Values for a certain Key.

3.4. Perspectives for the future development

The OSM software has a great perspective not only for analysis and assessment of bicycle infrastructure, but also both for local and global research different infrastructure data. There are a number of functions which concern future development, for example, function of "supernode" of simplified graph.

A number of next steps:

1. Function in development for calculation of area within official boundary
2. Spatial Analysis: future goals include
   a. Identify urban area
   b. Identify zones within urban area, e.g.,
      i. Shopping districts
      ii. Industrial areas
      iii. "Sleeping" area (quiet residential)
   c. Network analysis
      i. Quality of existing bike network
      ii. Generation of various ideal future networks
         1. Based on existing bikeways
         2. Envisioned freshly without considering existing bikeways

    3. Manual review by stakeholders and improvement of the model
3. Setting policy course through the tools and insights garnered above

4. Conclusions of Chapter 4

OpenStreetMap is project that is being used by people from all around the world. Its popularity shows its importance and necessity, but how everything OSM has own pluses and minuses.

The main advantage of the OSM project is its huge amount of free data. Every person may use the OpenStreetMap data for own aims keeping the copyright.

But the OpenStreetMap has own disadvantages. First of all, each user can edit the map and nobody will sure that added or changed data is correct.

Also, if user can't find some facility on the map, it doesn't mean that the facility is absent. It was not added on the map, may be. Simply put, nobody can be sure of map certainty.

The quality of the map in different parts is different. Mainly it depends on the country's state of development and the accessibility of the Internet. Theoretically, OpenStreetMap covers all Earth, but some countries have full datasets (Germany) and other only location of main roads and settlements (Somalia, Chad, North Korea).

In spite of own disadvantages, the popularity of the OpenStreetMap is increasing and the OSM has grown to 978, 483 (20.12.2012) contributors during 8 years. The overall conclusion is that the OpenStreetMap project is useful and important for people from around the world.

This analysis method was developed specifically for the BICY project to overcome problems with inconsistent or incomplete physical data. Basically, this analysis reads the GEO-referenced highways network of all partner cities from the OpenStreetMap server, including different kinds of cycling facilities. The unified highways attributes of OpenStreetMap allow a comparative analysis with the data collection of the currently implemented cycling infrastructure, received from partners. A geographical, network-oriented analysis in development allows a much deeper insight to the functionality of cycling infrastructure. Measures like connectivity, and the temporal competitiveness of the cycling network with respect to the road network, can be determined.

Discussions and Conclusions

1. Introduction

The key predictive indicator developed at this time has been the Cycling Index, which measures bikeways per capita (in km per person). When correlated to the percent bicycling in the survey-generated modal share, a strong linear relationship is observed:

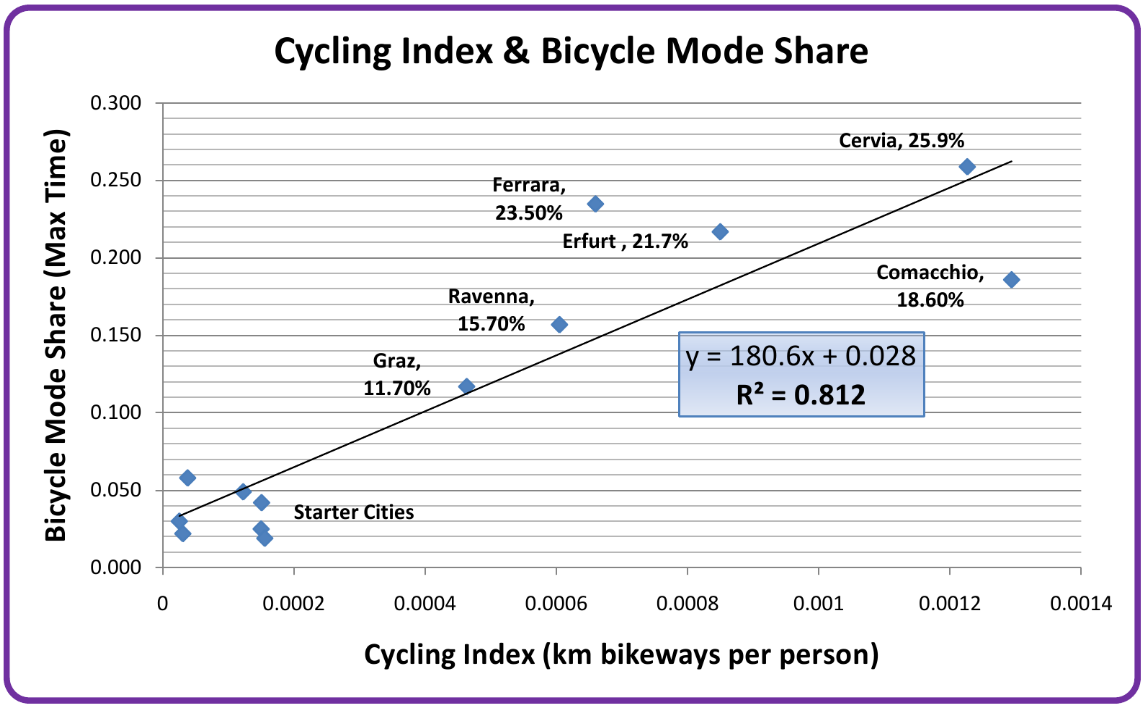


Figure 1: Cycling Index shown as it relates to Bicycle Mode Share (% cycling)

2. Bicycle Policy as implemention field for created methods

Here bicycle policies of two cities will be considered. There are Ravenna and Cervia.

Table 5.1 – Status Analysis

City	Ravenna	Cervia
Urban reference area	652,22	82,19

(km^2)		
Population(2008)	155597	28542
Males(number and % of population)	75982 (48,71%)	13639 (47,79%)
Females(number and % of population)	80015 (51,29%)	14903 (52,21%)
Children 8 and below(number and % of population)	14046 (9%)	2084 (7,3%)

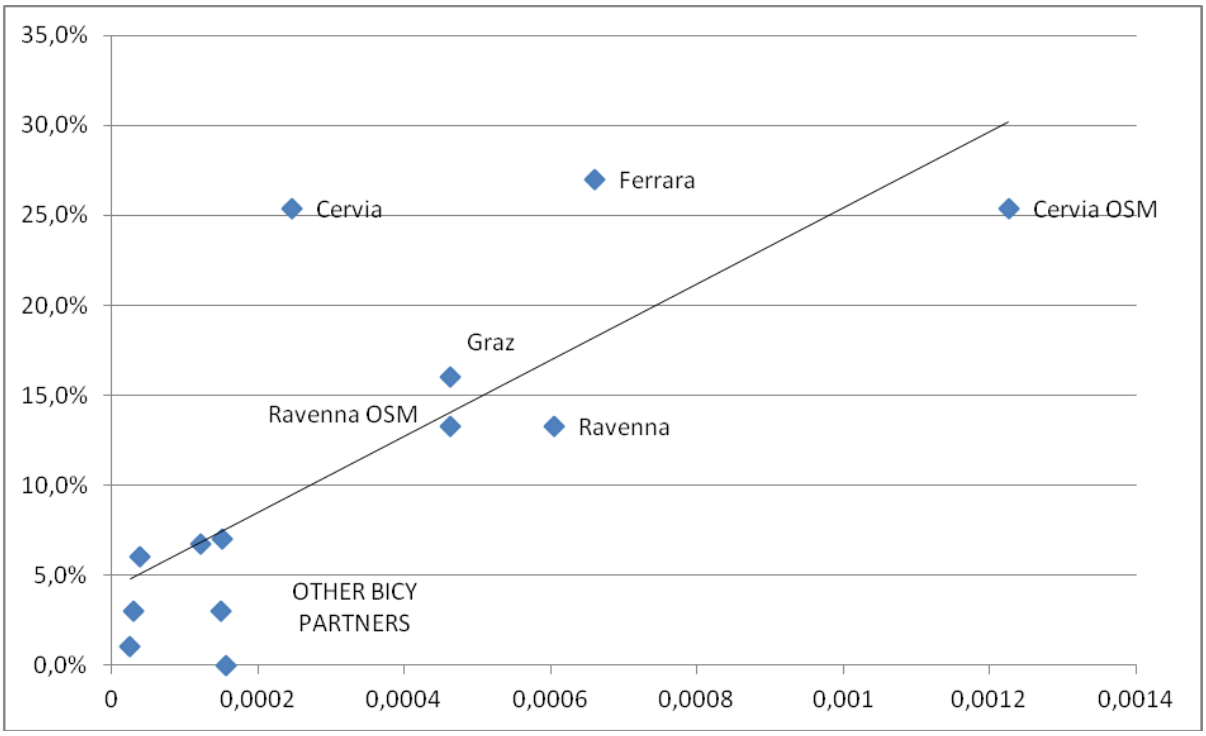
In both citys the urban reference area is not the true urban area from what the urban population density can be calculated as the real density is larger than 347,3 (Cervia) or 239,2 (Ravenna). Due to the fact that both cities have small urbanised areas around them which are connected with the center sometimes even by bicycle baths. So it would be a good idea to know exactly from which urban area and how many people enter these two cities every day.

Table 5.2 – Bicycle infrastructute

City	Ravenna	Cervia
Total road network (km)	919,47	Not given
Lenght of exclusive bikeways (km)	94,33	35
Bikeways according to OSM (km)	~ 43,39	62,55
Roads with mixed traffic/with traffic calming (km)	400	0,278
Pedestrian-only areas that allow bicycles to pass (m^2)	501	105,5

To get a better picture of the situation the length of the road network and bikeways were acquired from official sources. For verification purposes this data was compared with OSM which gave different results in both occations. The only thing is that OSM might not include all the recreation paths and for that reason both data was used in the analysis. But to know exactly what the exclusive bikeways are we need to define it. The exclusive bikeways are roads where bikes have a separated lane, either segregated by a line or by a physical separation like border, wall, flowerpots, etc. A bidirectional bike path on a single road counts only once. It is also important to know about roads which have mixed traffic and traffic calming if the research is done on bicycleways. So the mixed traffic roads are type of roads where motor vehicles and bicycles can drive and the traffic calming means the speedlimit is under 30 km/h . Pedestrian-only areas are places which allow bicycles to pass but this cannot be put into relation with road km.

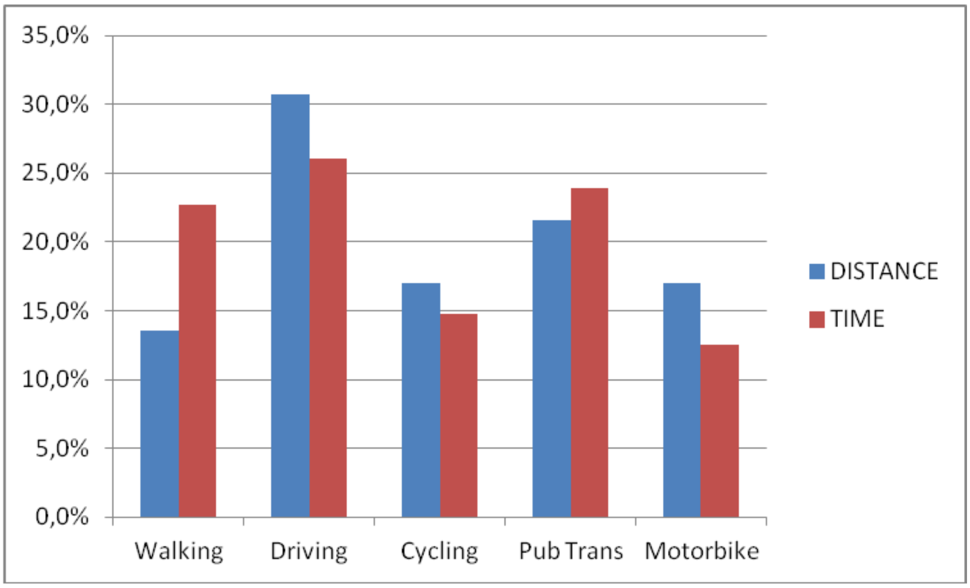
When making analysis the most standard indicators for level of bicycleways are cycling index which means how many km of cycling tracks there are per resident, network coverage index which means how many km cycling tracks there are comparing with the roads km and network density index which means how many km of cycling tracks there are in a km^2. These are the most standard indicators but road fatalities are also important when we look at the safety of the bicycleways. unfortunetly there is no data available for Cervia and data for Ravenna showed only severe injuries caused by cars which was 6,2 per 1000 persons.



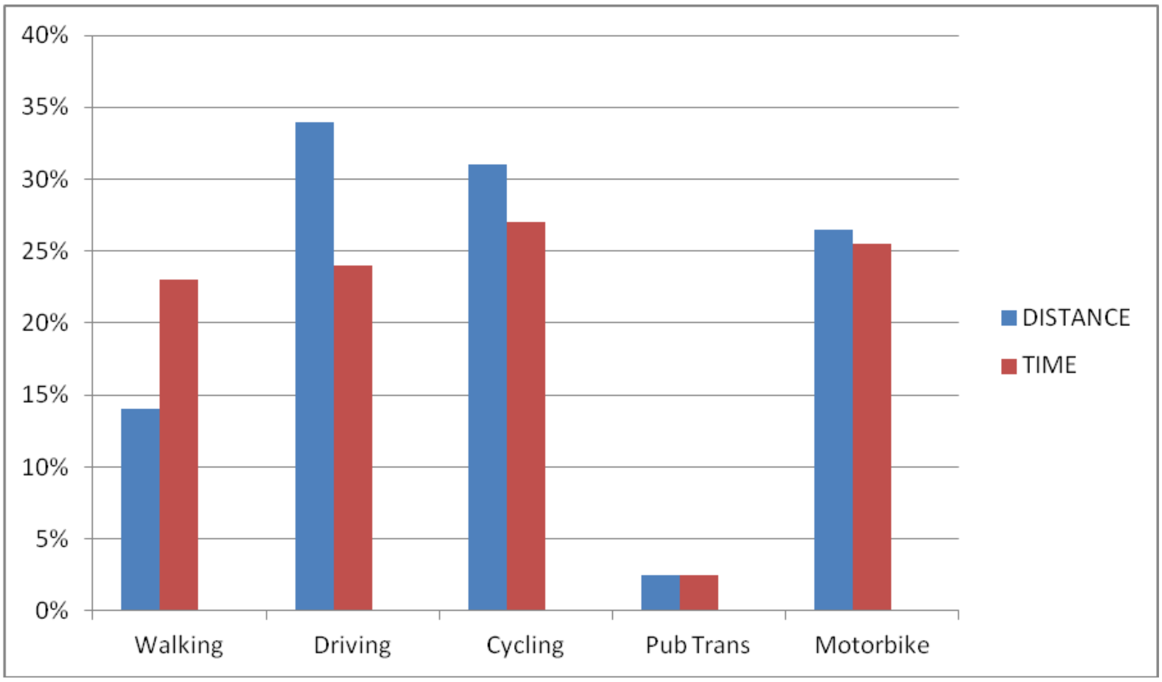
This shows the cycling index which was calculated twice using both OSM data and official data.

This graph suggests that if Ravenna wanted to increase its bike share to 30% then the length of separated bike lanes should grow by a factor of approximately 2.0. The result would be similar if the unadjusted bike share is used instead.

Since best way to get information what people think about the transportation and road network in some area is to conduct a survey. The survey was conducted in both cities but since it involved large overrepresentation of minor groups it is not as accurate as it could be thats why the following graphs do not take into account the age.



Here you can see the comparison of Ravennas distance bteween time with different transportation modes



Here you can see the comparison of Cervia distance bteween time with different transportation modes

Since the cycling is not very popular there are still things to do to improve the bicycleways and to do it in the way it would benefit the society most the conductors of the survey asked interviewees what would make them regular bicycle users. The results of the most common requested things are presented below:

City	Ravenna	Cervia

Safe bike parking	47.5%	15,2%
Continuous bike path	42.5%	31%
Bike path with weather protection	37%	20,3%
To take bike within public transport	30%	8,6%
Electric bikes	17%	5,6%
Bike-and-ride possibility	16%	5,6%
Bike sharing	10%	5,1%

To improve any situation you have to start with research and analysis. In Ravenna and Cervia conducted survey gave data that made possible to make SWOT analysis which helps us to find strengths, weaknesses, and external opportunities and threats. Also one very important part in this analysis was OSM as official sources to collect data about road and bicycleways length are not always plausible but it is essential to get exact data about these things to make accurate analysis. OSM gives us the best opurtunity to get data about urbanized area most accuretly.

Vision, Objectives and Targets

The potential is exceptional in both cities as in Ravenna 2/3 who cycle regularly do it in the rain and with cold(which means under 10 °C) and the same number in Cervia is 1/3. Not to mention 34,5% of people who do not cycle regularly are willing to do it if if certain requirements are met. The two most important things that need to be done is creating safe parking and continuous bike paths. There should be some work done on the safety of cyclists too as people would use bicycles more if they feel safe and respected on the road. Never the less both cities aim to be the best cycling city of it's size in the central europe. But to do that there are a lot of things that need to be done and taken into consideration as it is really easy to make mistakes that might end very badly for cyclists or discourage the use of bikes. Here we can see some points that need to be taken into consideration:

Ravenna	Cervia
Infrastructure targets	

Double the bikeways network to attain target cycling levels (Cycling Index > 0.0011)	Increase the bikeways network to attain maximum cycling levels
Assess the network and ensure maximum connectivity	Assess the network and ensure maximum connectivity
Assess the bikeway network based on the above guidance, and ensure it is high quality	Assess the bikeway network and ensure it is high quality
Sub-targets	
All bikeways are smooth	East-west connections
All bikeway connections are safe and obvious	Milano Marittima connection
All bikeways are wide enough to accommodate large numbers of cyclists of varied skill levels.	All bikeways are smooth
Bikeways are clearly for bicyclists, not used for walking or parking	All bikeway connections are safe and obvious
	All bikeways are wide enough to accommodate large numbers of cyclists of varied skill levels.
	Bikeways are clearly for bicyclists, not used for walking or parking

It is also very important that bicycle parking would be safe it means that the risk of theft would be minimum. In the other hand going everywhere by bike is not possible so for traveling longer distances it is necessary that the public transport has the capability to take bicycles inside and it has to be easy as everybody can not lift the bicycle. The same requirement has to be fulfilled with trains. Another thing that needs to be considered is laws and regulations since they need to assure the safety and not discouragement of the people who use the bike. But to increase the effect of laws on safety there should also be programs which teach drivers to respect the cyclers and ofcourse every child should be able to access the bicycle training in school. Similar program should be granted to elderly people. Both cities also think it would be good idea to make maps and other kind of information available for everybody.

In the future, based on the survey results, Ravenna wants to achieve minimum 30% cycling rate and Cervia wants to get at least 67,1%. It has many reasons why to increase cycling to this level. From persons perspective cycling is a active transport which reduces the health risks and people don't depend on other modes of transport any more which provides freedom of movement.

These plans are good but to ensure that everything is going according to the plan and to get a good picture of the situation there has to be done regular monitoring and evaluation. It is suggested that BYPAD audit should be done regularly after 2 or 3 years. It is also recommended to monitor the cycleways and parking places and ofcourse keep records of it. Counting the bicyclists every year or even every season will also give a good view on the situation and to make the information even more accurate it is recommended to do mobility survey after 5 years but the best way would be to do it yearly. And to encourage people more all this data and analysis should be made available for everybody even the data from GIS.

Action and Budget Plan

Since cycling is already existing mode of transport the funds should be found from the buget. The expected costs of bicycle lane per km is 22000€ an bicycle track 250000€ per km. For parking the cost of one outdoor bicycle rack which can hold 10 bicycles is 750€. But it is also necessary for residences and shops to provide adequate and safe storage for bicycles on the ground floor. If they can't provide it then the city should find the funds to do it. City also should also regularly monitor the situation, this includes the BYPAD audit and gathering injury and fatality data.

Idea to make bike sharing available in both cities would require some costs but everything depends on the complexity of the system. The simplest is when you get the bike to use you have to return it to the same place. That way there is no need of constant distribution and other activitys that need time and money. Here is a sample from Portland:

Smart Card bike sharing systems range in price from $4,500 - $5,500USD [per bike] including the cost of docking stations, computer software, licensing, bikes, and other capital expenditures. Operating costs range from $1250 - $2300USD/bike, although some business plans purport that over 50% of operating costs can be re-captured through subscriptions, rental and user fees. Cell phone activated systems tend to have a much lower capital costs ($1000-$2500/bike) than Smart Card systems, but require more staffing.

Assigning Responsibilities and Resources

To get maximum advantage all the resources like knowlege, funds and human resource must be used at the right time. This needs political involvement aswell but untill now Cervia is lacking in this department. That's why Cervia needs dedicated staff who would deal with this problem and also they need to meet all the requirements to improve the cycling situation it Cervia. Ideally expert should be hired to coordinate and look over all the work that is done and is still needed to be done to increase the efficiency.

Now if we look at Ravenna then they should establish bicycle office which could coordinate the different bicycle policys with needs and who would be a middle-man to get more support from municipality. Also coordinate all the improvements and changes done in this department. This is very important to do now when the investments are highest.

Monitoring and Evaluation

To see that efforts are done to improve the bikers situation in both cities the has to be done some monitoring. Closer look has to be taken into changes in infrastructure as this was the main problem in both cities. The monitoring has to be done simultaneously with all the infrastructure components so the investments and development can be measured.

To understand how the investments have worked it is also necessary to count the cyclists every year at the same time it is also suggested to do it with the same weather pattern to get the best data. The counting should be done in the busiest time.

When evaluating what is done it is very important what people think of the changes so it is necessary to conduct mobility survey at least once in 5 years and the data which is collected and created, including GIS datasets for analysis, should be made publicly available on the city’s website. This will improve the public’s knowledge and assist the city as a whole in attaining its important goals.

Conclusions

This first bicycle policy for both cities is a major step. It is the first of many more steps which must be taken. Quick and focused action is necessary, and funds must be allocated consistently, to achieve the great conditions and the many benefits possible.

References

1. White Paper (2001): *European transport policy for 2010: time to decide*, European Commission, Brussels
2. Communication from the Commission (2009): *A sustainable future for transport: Towards an integrated, technolo*
3. *gy-led and user friendly system*, European Commission, Brussels
4. White Paper (2011): *Roadmap to a Single European Transport Area – Towards a competitive and resource efficient transport system*, European Commission, Brussels
5. Transport 2050: Commission outlines ambitious plan to increase mobility and reduce emissions, http://europa.eu/rapid/press-release_IP-11-372_ga.htm (date: 10.01.2013, time: 17:16)
6. European Territorial Co-operation: http://ec.europa.eu/regional_policy/cooperate/cooperation/index_en.cfm
7. http://ec.europa.eu/regional_policy/cooperate/cooperation/transnational/index_en.cfm (date: 12.01.2013, time: 12:43)
8. Operational Programme CENTRAL EUROPA (2012), Version 2.1: http://www.central2013.eu/fileadmin/user_upload/Downloads/Document_Centre/Programme_Documents/OP_Central_Europe_approved_EC__2007_12_03.pdf (date: 12.01.2013, time: 14:01)
9. Project "BICY – Cities & Regions of Bicycles" approved by CENTRAL EUROPA: http://www.central2013.eu/nc/central-projects/approved-projects/funded-projects/?tx_fundedprojects_pi1%5Bproject%5D=56 (date: 12.01.2013, time: 14:02), pp.
10. Official website of the project "BICY – Cities & Regions of Bicycles": http://bicy.it/ (date: 12.01.2013, time: 14:04)
11. Karl Reiter, Susanne Wrighton, Christine Plank (2011), *Trendy cycling: 20 good reasons for cycling*, FGM-AMOR, page 2, http://bicy.it/docs/86/Trendy_cycling_EN_web.pdf (date: 12.01.2013, time: 15:17)
12. Work plan (2009): *Detailed work plan and timetable of activities per project partner*, BICY Project, Central Europe, ERDF
13. Joerg Schweizer, Jason N. Meggs (2012), *Final Identification of Common Indicators*, UNIBO-DICAM, BICY Project, Central Europe, ERDF

14. Joerg Schweizer, Jason Meggs, Anton Pashkevich (2011), Presentation on the 4th Transnational meeting: *BICY Workpackage 3 update*, UNIBO-DICAM, BICY Project, Central Europe, ERDF, Graz, Austria.
15. Joerg Schweizer (2010), BICY working paper for Task 3.2 Common Indicators: *Specification of data collection scheme*, Revision 01, UNIBO-DICAM, BICY Project, Central Europe, ERDF
16. CROW - Centre for Research and Contract Standardization in Civil And Traffic Engineering. Link: http://www.crow.nl/english
17. Pucher, John and Buehler, Ralph (2008) 'Making Cycling Irresistible: Lessons from The Netherlands, Denmark and Germany', Transport Reviews, 28:4, 495 — 528
18. Design Manual for bicycle traffic, CROW Record 25, Ede, NL
19. Ministry of Transport, Public Works and Water Management Directorate-General for Passenger Transport 2009, *Cycling in the Netherlands*, Ministry of Transport, Public Works and Water Management Directorate-General for Passenger Transport, Den Haag.
20. Fietsberaad 2006, *Continuous and integral: The cycling policies of Groningen and other European cycling cities*, Fietsberaad Publication No. 7, Amsterdam.
21. Rodney Tolley (ed.) (2003), *Sustaining sustainable transport: planning for walking and cycling in urban environments* , Cambridge (UK), Woodhead Publishing Limited, 507-524
22. Survey Guide: http://explorable.com/surveys-and-questionnaires (state: 14:02, 17.02.13)
23. Survey Research: http://www.socialresearchmethods.net/kb/survey.php (state: 14:05, 17.02.13)
24. Survey Research: http://edu-net.net/bus-writing/writing/guides/research/survey/index-2.html (state: 14:05, 17.02.13)
25. Survey: http://psychology.about.com/od/researchmethods/f/survey.htm (state: 14:20, 17.02.13)
26. Data Collection: http://people.uwec.edu/piercech/researchmethods/data%20collection%20methods/data%20collection%20methods.htm (state: 14:23, 17.02.13)
27. http://www.au.af.mil/au/awc/awcgate/navy/bpi_manual/mod7-datacoll.pdf (state: 14:31, 17.02.13)
28. Survey: http://psychology.about.com/od/researchmethods/f/survey.htm (state: 14:32, 17.02.13)
29. The main resource of OpenStreetMap: http://www.openstreetmap.org/

30. http://wiki.openstreetmap.org/wiki/Main_Page (state: 14:01, 19.12.12)
31. http://josm.openstreetmap.de/wiki/WikiStart (state: 16:20, 23.12.12)
32. http://wiki.openstreetmap.org/wiki/Merkaartor (state: 17:33, 23.12.12)
33. http://wiki.openstreetmap.org/wiki/Mapzen (state: 17:31, 23.12.12)
34. http://wiki.openstreetmap.org/wiki/Vespucci (state: 17:28, 23.12.12)
35. http://wiki.openstreetmap.org/wiki/Osm2go (state: 17:40, 23.12.12)
36. http://wiki.openstreetmap.org/wiki/Potlatch (state: 17:33, 23.12.12)
37. http://www.learnosm.org/ (state: 11:15, 4.01.13)
38. http://wiki.openstreetmap.org/wiki/Elements (state: 12:42, 7.01.13)
39. http://wiki.openstreetmap.org/wiki/Map_Features (state: 12:10, 8.01.13)
40. http://www.openstreetmap.org/copyright/ (state: 11:55, 9.01.13)
41. http://en.wikipedia.org/wiki/XML (state: 11:56, 9.01.13)
42. http://wiki.openstreetmap.org/wiki/OSM_XML (state: 11:56, 9.01.13)
43. http://wiki.openstreetmap.org/wiki/OSM_XML#OSM_XML_file_format (state: 11:58, 9.01.13)
44. http://wiki.openstreetmap.org/wiki/Downloading_data (state: 10:13, 10.01.13)
45. http://wiki.openstreetmap.org/wiki/Planet.osm (state: 10:20, 10.01.13)
46. http://planet.openstreetmap.org/ (state: 10:22, 10.01.13)
47. http://wiki.openstreetmap.org/wiki/Export (state: 11:17, 10.01.13)
48. http://wiki.openstreetmap.org/wiki/API (state: 11:20, 10.01.13)
49. http://wiki.openstreetmap.org/wiki/API_v0.6 (state: 11:25, 10.01.13)
50. http://wiki.openstreetmap.org/wiki/API_usage_policy (state: 11:27, 10.01.13)
51. http://www.itoworld.com/static/openstreetmap_tools.html (state: 12:31, 10.01.13)
52. http://en.wikipedia.org/wiki/OpenStreetMap (state: 17:43, 19.12.12)
53. http://wiki.openstreetmap.org/wiki/Way (state: 13:25, 7.01.13)
54. http://wiki.openstreetmap.org/wiki/Node (state: 13:31, 7.01.13)
55. http://wiki.openstreetmap.org/wiki/Relation (state: 13:38, 7.01.13)
56. http://creativecommons.org/licenses/by-sa/3.0/ (state: 15:12, 11.01.13)

Annexes

A. Tables of Data for Quantitative Analysis

Example of partner city Erfurt

Table 1

SURFACE FOR HIGHWAY TYPES

HIGHWAY TYPES	SURFACE TYPES	FEATURES OF SURFACE TYPES and EXPLANATIONS	VALUE	UNIT
motorway		TOTAL LENGTH	1562	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	1562	[meters]
motorway_link		TOTAL LENGTH	4947	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	4947	[meters]
trunk		TOTAL LENGTH	20939	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	20939	[meters]
trunk_link		TOTAL LENGTH	6498	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	6498	[meters]
primary		TOTAL LENGTH	15301	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	15301	[meters]
primary_link		TOTAL LENGTH	52	[meters]

	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	52	[meters]
secondary		TOTAL LENGTH	62748	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	62748	[meters]
secondary_link		TOTAL LENGTH	51	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	51	[meters]
tertiary		TOTAL LENGTH	41821	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	1674	[meters]
		concrete	279	[meters]
		Total Paved Surface	1953	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	1953	[meters]
		unclassified	39868	[meters]
tertiary_link		TOTAL LENGTH	11	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	11	[meters]
residential		TOTAL LENGTH	231659	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	18319	[meters]
		paved	4639	[meters]
		cobblestone	9731	[meters]
		Total Paved Surface	32689	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		unpaved	773	[meters]
		gravel	352	[meters]
		ground	203	[meters]
		Total Unpaved Surface	1328	[meters]
	UNKNOWN SURFACE	Separate for every value and in total:		
		bricks	21	[meters]
		Total Unknown Surface	21	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		

		classified	34038	[meters]
		unclassified	197621	[meters]
unclassified		TOTAL LENGTH	35706	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	920	[meters]
		paved	766	[meters]
		concrete	767	[meters]
		Total Paved Surface	2453	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		unpaved	1977	[meters]
		compacted	776	[meters]
		Total Unpaved Surface	2753	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	5206	[meters]
		unclassified	30500	[meters]
road		TOTAL LENGTH	225	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	225	[meters]
living_street		TOTAL LENGTH	32911	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	905	[meters]
		paved	2536	[meters]
		cobblestone	4491	[meters]
		paving_stones	807	[meters]
		Total Paved Surface	8739	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		unpaved	42	[meters]
		Total Unpaved Surface	42	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	8781	[meters]
		unclassified	24130	[meters]
service		TOTAL LENGTH	150706	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	10517	[meters]
		paved	8356	[meters]
		cobblestone	1486	[meters]
		concrete	374	[meters]
		paving_stones	326	[meters]
		Total Paved Surface	21059	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		dirt	71	[meters]
		unpaved	2399	[meters]
		gravel	253	[meters]
		compacted	459	[meters]

		ground	231	[meters]
		sand	352	[meters]
		Total Unpaved Surface	3765	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	24824	[meters]
		unclassified	125882	[meters]
track		TOTAL LENGTH	102483	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	10909	[meters]
		paved	1471	[meters]
		cobblestone	646	[meters]
		concrete	3095	[meters]
		paving_stones:50	53	[meters]
		Total Paved Surface	16174	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		grass	9788	[meters]
		dirt	1001	[meters]
		unpaved	7436	[meters]
		gravel	9032	[meters]
		compacted	1714	[meters]
		ground	7991	[meters]
		earth	80	[meters]
		Total Unpaved Surface	37042	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	53216	[meters]
		unclassified	49267	[meters]
pedestrian		TOTAL LENGTH	21521	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	49	[meters]
		paved	825	[meters]
		cobblestone	1616	[meters]
		paving_stones	1448	[meters]
		Total Paved Surface	3938	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	3938	[meters]
		unclassified	17583	[meters]
footway		TOTAL LENGTH	179910	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	1182	[meters]
		paved	5814	[meters]
		cobblestone	1380	[meters]
		paving_stones:30	90	[meters]
		concrete	16	[meters]
		paving_stones	6879	[meters]
		paving_stones:25	144	[meters]
		Total Paved Surface	15505	[meters]

	UNPAVED SURFACE	Separate for every value and in total:		
		unpaved	5667	[meters]
		gravel	2328	[meters]
		compacted	7	[meters]
		ground	60	[meters]
		wood	41	[meters]
		pebblestone	292	[meters]
		Total Unpaved Surface	8395	[meters]
	UNKNOWN SURFACE	Separate for every value and in total:		
		iron	32	[meters]
		Total Unknown Surface	32	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	23932	[meters]
		unclassified	155978	[meters]
path		TOTAL LENGTH	62163	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	798	[meters]
		paved	1535	[meters]
		cobblestone	459	[meters]
		paving_stones:30	159	[meters]
		concrete	80	[meters]
		paving_stones:50	12	[meters]
		paving_stones	6750	[meters]
		Total Paved Surface	9793	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		grass	1743	[meters]
		unpaved	1043	[meters]
		gravel	8366	[meters]
		compacted	27	[meters]
		ground	5676	[meters]
		sand	105	[meters]
		earth	584	[meters]
		fine_gravel	449	[meters]
		Total Unpaved Surface	17993	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	27786	[meters]
		unclassified	34377	[meters]
cycleway		TOTAL LENGTH	46012	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		asphalt	5653	[meters]
		paved	3379	[meters]
		cobblestone	185	[meters]
		paving_stones:30	593	[meters]
		concrete	6096	[meters]
		paving_stones:50	153	[meters]
		paving_stones	16235	[meters]
		paving_stone:120	385	[meters]

		Total Paved Surface	32679	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		gravel	834	[meters]
		compacted	2225	[meters]
		Total Unpaved Surface	3059	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	35738	[meters]
		unclassified	10274	[meters]
steps		TOTAL LENGTH	6247	[meters]
	PAVED SURFACE	Separate for every feature and in total:		
		paved	77	[meters]
		cobblestone	205	[meters]
		concrete	15	[meters]
		paving_stones	6	[meters]
		Total Paved Surface	303	[meters]
	UNPAVED SURFACE	Separate for every value and in total:		
		unpaved	171	[meters]
		wood	18	[meters]
		Total Unpaved Surface	189	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	492	[meters]
		unclassified	5755	[meters]
platform		TOTAL LENGTH	3271	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	3271	[meters]
proposed		TOTAL LENGTH	604	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	604	[meters]
construction		TOTAL LENGTH	219	[meters]
	PAVED SURFACE	No information about paved surface	-	[meters]
	UNPAVED SURFACE	No information about unpaved surface	-	[meters]
	SURFACE IN COMMON	SURFACE is classified/unclassified:		
		classified	0	[meters]
		unclassified	219	[meters]

B. Module “OpenStreetLib”

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
OpenStreetLib module
"""
__author__ = """Anton Pashkevich"""

##########################################################################################
# import of necessary libraries

import networkx as nx
from os import path
import urllib
from xml.parsers import expat
import math
import pickle
import numpy as np

from lib.lib_graphics import *
from lib.lib_assignment import *

from intersection import *
from intersection_v2 import *

##########################################################################################
# definition of necessary lists of OSM Features

ROADATTRS =
['motorway','motorway_link','trunk','trunk_link','primary','primary_link','secondary','secondary_link',
'tertiary', 'tertiary_link' ,'residential','unclassified','road','living_street','service','track']
MAINFEATURES = ['aerialway', 'aeroway', 'amenity', 'boundary', 'craft', 'geological', 'highway',
'historic', 'leisure', 'man_made', 'military', 'natural', 'office', 'place', 'power', 'railway', 'route', 'shop',
'sport', 'tourism', 'waterway', 'barrier', 'building', 'emergency', 'landuse', 'public_transport']
IDLIB = {'aerialway': 'idalw', 'aeroway': 'idaw', 'amenity': 'idam', 'barrier': 'idba', 'boundary': 'idbo',
'building': 'idbu', 'craft': 'idcr', 'emergency': 'idem', 'geological': 'idge', 'highway': 'idhw', 'historic':
'idhi', 'landuse': 'idlu', 'leisure': 'idle', 'man_made': 'idmm', 'military': 'idmi', 'natural': 'idna', 'office':
'idof', 'place': 'idpl', 'power': 'idpo', 'public_transport': 'idpt', 'railway': 'idrw', 'route': 'idro', 'shop':
'idsh', 'sport': 'idsp', 'tourism': 'idto', 'waterway': 'idww'}
IDLIST = ['idalw', 'idaw', 'idam', 'idba', 'idbo', 'idbu', 'idcr', 'idem', 'idge', 'idhw', 'idhi', 'idlu', 'idle',
'idmm', 'idmi', 'idna', 'idof', 'idpl', 'idpo', 'idpt', 'idrw', 'idro', 'idsh', 'idsp', 'idto', 'idww', 'id']
PAHTATTRS = []
CYCLEWAYS = ['cycleway','cycleway:left', 'cycleway:right']
TYPE_CYCLEWAYS = ['lane', 'track', 'opposite_lane', 'opposite_track', 'opposite', 'shared',
'share_busway', 'opposite_share_busway', 'shared_lane', 'sharrow', 'yes'] # 'yes' - it is not in the list
BICYCLES = ['yes', 'designated', 'destination', 'permissive', 'official', 'unknown']
```

```
SURFACE = ['paved', 'unpaved', 'asphalt', 'concrete', 'paving_stones', 'cobblestone', 'metal', 'wood',
'grass_paver', 'gravel', 'pebblestone', 'grass', 'ground', 'earth', 'dirt', 'mud', 'sand']
PAVED = ['asphalt', 'cobblestone', 'cobblestone:flattened', 'concrete', 'concrete:lanes',
'concrete:plates', 'paving_stones', 'paving_stones:30', 'paving_stones:20', 'paving_stones:50',
'paving_stones:25', 'paving_stone:120', 'sett', 'tartan']
UNPAVED = ['artificial_turf', 'compacted', 'dirt', 'earth', 'fine_gravel', 'grass', 'grass_paver', 'gravel',
'ground', 'metal', 'mud', 'pebblestone', 'sand', 'wood', 'clay']

##################################################################################
# identification of class "Processing" - class for parsing XML-files

class Processing:

  def __init__(self):
    self.t = 0 # idea requires completion
    self.s = 0 # idea requires completion
    self.h = 0 # idea requires completion
    self.node = [] # dictionary of nodes
    self.way = [] # dictionary of ways
    self.relation = [] # dictionary of relations
    self.count_ot = [] # counter of XML open tags
    self.count_et = [] # counter of XML end tags

  def start_element(self, name, attrs):
    self.open_tag = name
    self.count_ot.append(name) # counter of open tags (ot) - len(self.count_ot)
    a = attrs
    self.b = attrs
    if name == 'node': # formatting of dictionary 'node'
      if a['visible'] == 'true':
        if a.has_key('timestamp'):
          del a['timestamp']
        if a.has_key('uid'):
          del a['uid']
        if a.has_key('user'):
          del a['user']
        if a.has_key('version'):
          del a['version']
        if a.has_key('changeset'):
          del a['changeset']
        del a['visible']
        a['lat'] = float(a['lat'])
        a['lon'] = float(a['lon'])
        a['id'] = int(a['id'])
        self.node.append(a)

    if name == 'way': # formatting of dictionary 'way'
      if a['visible'] == 'true':
        if a.has_key('timestamp'):
          del a['timestamp']
        if a.has_key('uid'):
          del a['uid']
        if a.has_key('user'):
```

```
                del a['user']
            if a.has_key('version'):
                del a['version']
            if a.has_key('changeset'):
                del a['changeset']
            del a['visible']
            a['id'] = int(a['id'])
            a['coord'] = []
            self.way.append(a)
            self.t = 1
            self.e = 0 # 04.03.2012

    if name == 'relation': # formatting of dictionary 'relation'
        if a['visible'] == 'true':
            if a.has_key('timestamp'):
                del a['timestamp']
            if a.has_key('uid'):
                del a['uid']
            if a.has_key('user'):
                del a['user']
            if a.has_key('version'):
                del a['version']
            if a.has_key('changeset'):
                del a['changeset']
            del a['visible']
            a['id'] = int(a['id'])
            self.relation.append(a)
            self.s = 1

def end_element(self, name):
    self.end_tag = name
    self.count_et.append(name) # counter of end tags (et) - len(self.count_et)
    if len(self.count_ot)-len(self.count_et) == 2:
        # write all nodes of WAY
        if name == 'nd':
            if self.count_ot[-2] == 'way':
                self.coord = []
                self.coord.append(int(self.b['ref']))
                self.way[-1]['coord'] = self.coord
                self.n = 1
                self.m = 1
            if self.count_ot[-2-self.m] == 'way':
                if self.coord[self.n-1] == int(self.b['ref']):
                    pass
                else:
                    self.coord.append(int(self.b['ref']))
                    self.way[-1]['coord'] = self.coord
                    self.n = self.n+1
                    self.m = self.m+1

        # write all members of RELATION
        if name == 'member': # it requires improvement
            if self.count_ot[-2] == 'relation':
```

```
            self.memb = []
            self.b['ref'] = int(self.b['ref'])
            self.memb.append(self.b)
            self.relation[-1]['members'] = self.memb
            self.y = 1
            self.z = 1
        if self.count_ot[-2-self.z] == 'relation':
            self.b['ref'] = int(self.b['ref'])
            if self.memb[self.y-1] == self.b:
                pass
            else:
                self.memb.append(self.b)
                self.relation[-1]['members'] = self.memb
                self.y = self.y+1
                self.z = self.z+1

    if name == 'tag': # write tags of NODE
        if self.count_ot[-2] == 'node':
            self.node[-1][self.b['k']] = self.b['v']
            self.h = 1
        #else:
        #    self.h = 0# - last 3.03.2012
        if self.count_ot[-2-self.h] == 'node':
            # there is one small nuance:
            # in some occasions write information two times
            # REWRITING the information from first cycle
            self.node[-1][self.b['k']] = self.b['v']
            self.h = self.h+1

    if name == 'tag' and self.t == 1: # write tags of WAY
        if self.count_ot[-2] == 'nd':
            self.way[-1][self.b['k']] = self.b['v']
            self.e = 1
        #else:
        #    self.e = 0# - last 3.03.2012
        if self.count_ot[-2-self.e] == 'nd':
            # the same - REWRITING
            self.way[-1][self.b['k']] = self.b['v']
            self.e = self.e+1

    if name == 'tag' and self.s == 1: # write tags of RELATION
        if self.count_ot[-2] == 'member':
            self.relation[-1][self.b['k']] = self.b['v']
            self.o = 1
        #else:
        #    self.o = 0# - last 3.03.2012
        if self.count_ot[-2-self.o] == 'member':
            # the same - REWRITING
            self.relation[-1][self.b['k']] = self.b['v']
            self.o = self.o+1

##########################################################################################
# Functions for processing OpenStreetMap
```

```
####################################################################################
# 1 - functions for Writing/Reading information getting from OpenStreetMap


def save_graf_as_txt(graf, filename=None, attr = None, value=None):
    """
    Information about the created graf is saved in txt-file
    """
    if filename == None:
      filename = "graf-boundary.txt"

    if attr == None:
        attr = 'boundary'

    graf_file = open(filename, 'w')

    G = get_graf_one_key_clean(graf, attr)
    #G = get_graf_one_key(graf, attr)
    #G = get_graf_one_value_clean(graf, attr, value)

    print 'Start'

    for node in G.nodes():
        graf_file.write(str(node)+" ")
        graf_file.write(str(G.node[node]))
        graf_file.write("\n")

    id_edge = []
    for edge in G.edges():
        i,j = edge
        for n in range(G[i][j]['edge_numbers']):
            if id_edge.count(G.edge[i][j]['id'+str(n+1)]) == 0:
                id_edge.append(G.edge[i][j]['id'+str(n+1)])

    print 'Middle'

    for ide in id_edge:
      for edge in G.edges():
            i,j = edge
            ok = 0
            for n in range(G[i][j]['edge_numbers']):
          if G.edge[i][j]['id'+str(n+1)] == ide:
              ok = 1
            if ok == 1:
              graf_file.write(str(ide)+" ")
              graf_file.write(str(edge)+" ")
              graf_file.write(str(G.edge[i][j]))
              graf_file.write("\n")

    print 'End'

    graf_file.close
```

```
    return G



def save_graf_as_pickle(graf, filename=None):
    """
    Created graf is saved as pickle-file
    """
    if filename == None:
      filename = "graf.dat"

    output_file = open(filename, 'w')
    p1 = pickle.Pickler(output_file) #, protocol = 2) # relevant UNIX-features
    p1.dump(graf)
    output_file.close()



def get_graf_from_pickle(filename):
    """
    Get graf from pickle-file
    """
    file_graf = open(filename, "r")

    graf = pickle.load(file_graf)

    file_graf.close()

    return graf


############################################################################################
# 2 - functions for processing XML-files: parsing, getting and saving OSM data


def get_graf(coords, graf=None, cityname=None):
    """
    Reads xml-file from openstreetmap into graf
    Variable "coord" is a list of constants
    """
    if graf == None:
      graf = nx.Graph()

    tiles, nlength, nwidth = get_tiles_from_square(coords)

    if cityname == None:
      cityname = 'city'
    get_xml_files(tiles, nlength, nwidth, cityname)


    G = graf
    for y in range(nwidth+1):
      if y > 0:
```

```
        for x in range(nlength+1):
         if x > 0:
          filename = cityname + str(y) + '-' + str(x) + '.xml'
          """CHECK UP"""
          #print filename
          G = get_graf_from_xml_direct(filename, graf=G)

  H = square_boundary(G, coords)
  return H



def get_tiles_from_square(coords):
  """
  Divide the piece of map into tiles and
  Return the list of coordinates for all tiles
  DON'T PAY ATTENTION TO MINUS
  """

  # prefix l for longitude
  # prefix w for latitude

  tiles = []
  main_minlong,main_minlat,main_maxlong,main_maxlat = coords

  # Constant values of small-size pieces (in accord with the city Groningen)
  const_l = 0.0315
  const_w = 0.0152
  safety = 0.0001

  width_coord = main_maxlat - main_minlat
  length_coord = main_maxlong - main_minlong


  int_w = int(width_coord/const_w)
  int_l = int(length_coord/const_l)

  rest_w = width_coord - const_w*int_w
  rest_l = length_coord - const_l*int_l

  if rest_w > 0:
   nwidth = int_w + 1
  else:
   nwidth = int_w

  if rest_l > 0:
   nlength = int_l + 1
  else:
   nlength = int_l

  # Quantity of cells:
  # quantity of columns = int_l+1
  # quantity of rows = int_w+1
```

```
# dictionaries of coordinates
l = range(int_l+2)
w = range(int_w+2)

# Generation of coordinates for small-size pieces
for i in range(int_w+2):
        if i == 0:
           w[i] = main_minlat
        elif i == int_w + 1:
           w[i] = main_minlat + (i-1)*const_w + rest_w
    else:
       w[i] = main_minlat + (i)*const_w

for j in range(int_l+2):
        if j == 0:
           l[j] = main_minlong
        elif j == int_l+1:
           l[j] = main_minlong + (j-1)*const_l + rest_l
        else:
           l[j] = main_minlong + (j)*const_l

"""CHECK UP"""
print "coordinates of longitude l = ", l
print "coordinates of latitude w = ", w

# Generation of  tiles coordinates list

for i in range(int_w+2):
    if i > 0:
       if i == int_w+1:
          for j in range(int_l+2):
             if j > 0:
                if j == int_l+1:
                   """CHECK UP"""
                   #print 'i=', i, 'j=', j
                   tiles.append(((l[j-1]-safety),(w[i-1]-safety),(l[j]),(w[i])))
                              else:
                   """CHECK UP"""
                   #print 'i=', i, 'j=', j
                   tiles.append(((l[j-1]),(w[i-1]-safety),(l[j]+safety),(w[i])))
       else:
          for j in range(int_l+2):
             if j > 0:
                if j == int_l+1:
                   """CHECK UP"""
                   #print 'i=', i, 'j=', j
                   tiles.append(((l[j-1]-safety),(w[i-1]),(l[j]),(w[i]+safety)))
                else:
                   """CHECK UP"""
                   #print 'i=', i, 'j=', j
                   tiles.append(((l[j-1]),(w[i-1]),(l[j]+safety),(w[i]+safety)))
```

```
    return tiles, nlength, nwidth



def get_graf_from_tile(tile, graf=None):
    """
    Read tile from website - openstreetmap.org
    Only nodes and ways are processed
    Create a graf of this nodes and ways
    """

    if graf == None:
      graf = nx.Graph()

    lon,lat,mlon,mlat = tile
    link=
'http://api.openstreetmap.org/api/0.6/map?bbox='+str(lon)+','+str(lat)+','+str(mlon)+','+str(mlat)
    filename = 'data-'+'city'+'-temp_x.xml'
    # Need a modernization in 'filename' for saving all the XML-files fo large map piece
    urllib.urlretrieve(link,filename)

    xml_file = open('data-city-temp_x.xml','r')

    # Commands of XML-file parsing
    pr = Processing()
    p = expat.ParserCreate()
    p.returns_unicode = 0
    p.StartElementHandler = pr.start_element
    p.EndElementHandler = pr.end_element

    # Parsing of XML-file
    p.ParseFile(xml_file)


    # Addition new elements to graph

    # NEW NODES
    for node in pr.node:
      id_node = node['id']
      coords = node['lon'], node['lat']
      keys = node.keys()
      if graf.has_node(id_node):
            pass
      else:
            graf.add_node(id_node, pos = coords)
            for key in keys:
              if key == 'id' or key == 'lon' or key == 'lat':
                pass
              else:
                graf.node[id_node][key] = node[key]

    # NEW EDGES
    # get this part from "get_graf_from_xml_direct(filename, graf=None)"
```

```
   for way in  pr.way:
     ids_node = way['coord']
     id1 = way['id']
     for i in xrange(len(ids_node)):
             if i>0:
               edge = (ids_node[i-1], ids_node[i])
               if graf.has_edge(ids_node[i-1], ids_node[i]):
                 id2 = graf.edge[ids_node[i-1]][ids_node[i]]['id']
                 if id1 == id2:
                   pass
                 else:
                   k = int(str(id1)+str(i-1))
                   graf.add_node(k, pos = graf.node[ids_node[i-1]]['pos'])
                   g = int(str(id1)+str(i))
                   graf.add_node(g, pos = graf.node[ids_node[i]]['pos'])
                   graf.add_edges_from([ (k, g, way),])
                   """CHECK UP"""
                   #print k, g, '\n'
                   del graf[k][g]['coord']
               else:
                 graf.add_edges_from([ (ids_node[i-1], ids_node[i], way),])
                 del graf[ids_node[i-1]][ids_node[i]]['coord'] # 'id' is the same for all edges in one way

   return graf



def get_xml_files(tiles, nlength, nwidth, cityname):
   """
   Read tiles from website - openstreetmap.org - and save them as xml-files
   """
   y = 1
   x = 1

   for tile in tiles:
            print 'tile', tile
            if y <= nwidth:
              if x<= nlength:
             lon,lat,mlon,mlat = tile
                        link                                                                                             =
'http://api.openstreetmap.org/api/0.6/map?bbox='+str(lon)+','+str(lat)+','+str(mlon)+','+str(mlat)
                        filename = cityname+str(y)+'-'+str(x)+'.xml'
                        urllib.urlretrieve(link, filename)
                        x = x + 1
                        """CHECK UP"""
                        #print 'tile', tile
                        if x-1 == nlength:
                          x = 1
                          y = y + 1



def get_graf_from_xml_direct(filename, graf=None):
```

```
"""
Return graf reading directly from XML-file
"""

if graf == None:
   graf = nx.Graph()

xml_file = open(filename,'r')

# Commands of XML-file parsing
pr = Processing()
p = expat.ParserCreate()
p.returns_unicode = 0
p.StartElementHandler = pr.start_element
p.EndElementHandler = pr.end_element

# Parsing of XML-file
p.ParseFile(xml_file)


# Addition new elements to graph

# NEW NODES
for node in pr.node:
  id_node = node['id']
  coords = node['lon'], node['lat']
  keys = node.keys()
  if graf.has_node(id_node):
        pass
  else:
        graf.add_node(id_node, pos = coords)
        for key in keys:
          if key == 'id' or key == 'lon' or key == 'lat':
            pass
          else:
            graf.node[id_node][key] = node[key]

# NEW EDGES

for way in  pr.way:
  ids_node = way['coord']

  waykeys = way.keys()
  waykeys.remove('coord')


  ref1 = ids_node[0]
  ok = 0
  for ref in ids_node:
      if ok > 0:
          ref2 = ref

          if graf.has_edge(ref1,ref2) == True:
```

```
            contrik = 0
            for i in range(graf[ref1][ref2]['edge_numbers']):
              if graf[ref1][ref2]['id'+str(i+1)] == way['id']:
                contrik = 1
                """CHECK UP"""
                #print "repeated ways"
                #print graf[ref1][ref2]
                #print ref1, ref2
                #print way

            if contrik == 0:
              graf[ref1][ref2]['edge_numbers'] = graf[ref1][ref2]['edge_numbers'] + 1
              graf[ref1][ref2]['lib'+str(graf[ref1][ref2]['edge_numbers'])] = {}
              for i in waykeys:
                graf[ref1][ref2]['lib'+str(graf[ref1][ref2]['edge_numbers'])][i] = way[i]
                if i == 'id':
                  graf[ref1][ref2]['id'+str(graf[ref1][ref2]['edge_numbers'])] = way['id']
                else:
                  if graf[ref1][ref2].has_key(i) == False:
                    graf[ref1][ref2][i] = way[i]

          else:
            graf.add_edges_from([ (ref1, ref2, way),])
            del graf[ref1][ref2]['coord']
            graf[ref1][ref2]['id1'] = graf[ref1][ref2]['id']
            del graf[ref1][ref2]['id']
            graf[ref1][ref2]['lib1'] = {}
            for i in waykeys:
              graf[ref1][ref2]['lib1'][i] = way[i]
            graf[ref1][ref2]['edge_numbers'] = 1

          ref1 = ref

        ok = ok + 1

    return graf


##########################################################################################
# 3 - functions for processing of OSM data


def get_len(graf,edge):
    """
    Return length of given edge
    """

    n1, n2 = edge
    llong1, llat1 = graf.node[n1]['pos']
    llong2, llat2 = graf.node[n2]['pos']

    #pi - value of pi, rad - the radius of sphere (Earth)
    rad = 6372795
```

```
    #coordinates of two points
    #llat1 = 77.1539
    #llong1 = -120.398

    #llat2 = 77.1804
    #llong2 = 129.55

    #in radians
    lat1 = llat1*math.pi/180.
    lat2 = llat2*math.pi/180.
    long1 = llong1*math.pi/180.
    long2 = llong2*math.pi/180.

    #cosines and sines of latitudes and difference of longitude
    cl1 = math.cos(lat1)
    cl2 = math.cos(lat2)
    sl1 = math.sin(lat1)
    sl2 = math.sin(lat2)
    delta = long2 - long1
    cdelta = math.cos(delta)
    sdelta = math.sin(delta)

    #calculation of big circle length
    y = math.sqrt(math.pow(cl2*sdelta,2)+math.pow(cl1*sl2-sl1*cl2*cdelta,2))
    x = sl1*sl2+cl1*cl2*cdelta
    ad = math.atan2(y,x)

    dist = round(ad*rad, 2)

    return dist



def get_len_two_node(graf,node1,node2,weight=None):
    """
    Return the shortest lenght (in metes) between two nodes in graf
    """

    lenght = 0

    if weight == None:
        try:
            path = nx.dijkstra_path(graf,node1,node2)
        except nx.NetworkXError:
            #raise nx.NetworkXError ('Problem')
            lenght = 100000000
    else:
        try:
            path = nx.dijkstra_path(graf,node1,node2,weight=weight)
        except nx.NetworkXError:
            #raise nx.NetworkXError ('Problem')
            lenght = 100000000
```

```
    if lenght == 0:
        count = 0
        for i in path:
            count = count+1
            end = i
            if count > 1:
                stuck = get_len(graf, (start, end))
                lenght = lenght+stuck
            start = i

    return lenght



def is_intersected(x11,y11,x12,y12,x21,y21,x22,y22):
    """
    Takes two  lines  represented by the two extremities x11,y11,x12,y12
    and x21,y21,x22,y22.
    The result is  bool variable and true if both lines intersect.
    Line parameters can also be given in vectorial format where each
    row corrisponds to a line.
    """

    dx1 = x12-x11
    dy1 = y12-y11
    dx2 = x22-x21
    dy2 = y22-y21

    length1 = np.sqrt(dx1**2+dy1**2)
    length2 = np.sqrt(dx2**2+dy2**2)
    #print '  length1=',length1
    #print '  length2=',length2
    mx1 = dx1/length1
    mx2 = dx2/length2
    my1 = dy1/length1
    my2 = dy2/length2
    #print '  mx1 = ',mx1
    #print '  mx2 = ',mx2
    #print '  my1 = ',my1
    #print '  my2 = ',my2
    la1=-(mx2*(y11-y21)+my2*x21-my2*x11)/(mx2*my1-mx1*my2)
    la2=-(mx1*(y11-y21)+my1*x21-my1*x11)/(mx2*my1-mx1*my2)

    #print '  la1 =',la1
    #print '  la2 =',la2

    #x = x11+la1*mx1
    #y = y11+la1*my1
    #print '  x=',x
    #print '  y=',y

    #x = x21+la2*mx2
```

```
    #y = y21+la2*my2
    #print '  x=',x
    #print '  y=',y

    return  (la1<=length1)& (la2<=length2)



def r_return(x,x1,x2):
    """
    Help-application for function "is_intersected"
    """

    r = 0

    if x1 <= x2:
        if (x1 <= x) and (x <= x2):
            r = 1
    else: # x11 > x12
        if (x2 <= x) and (x <= x1):
            r = 1

    return r



def supernode_coord(nodelist, IDjunction):
    """
    Return coordinats of supernode used
    'Modifiziertes Miehle-Verfahren fuer Standortplanung'
    """

    epsilon = 0.00000001
    alfa = 0.000000000000001

    IDnodelib = []

    Xcoord = []
    Ycoord = []

    WeightValue = []


    for i in nodelist:
        id_node, coord, wv = i
        x, y = coord
        IDnodelib.append(id_node)
        Xcoord.append(x)
        Ycoord.append(y)
        WeightValue.append(wv)

    WVsum = 0
    WXsum = 0
```

```
WYsum = 0

for i in xrange(len(IDnodelib)):
   WVsum = WVsum+WeightValue[i]
   WXsum = WXsum+Xcoord[i]*WeightValue[i]
   WYsum = WYsum+Ycoord[i]*WeightValue[i]

# Graviation coordinates
x0 = WXsum/WVsum
y0 = WYsum/WVsum

x1 = x0
y1 = y0

stop = 0 # 0 - No, 1 - Yes

while stop != 1:
   constlist = []
   for i in xrange(len(IDnodelib)):
      const = math.sqrt((x1-Xcoord[i])**2+(y1-Ycoord[i])**2+epsilon)
      constlist.append(const)

   x2up = 0
   y2up = 0
   downpart = 0
   for i in xrange(len(IDnodelib)):
      x2up = x2up+(WeightValue[i]*Xcoord[i]/constlist[i])
      y2up = y2up+(WeightValue[i]*Ycoord[i]/constlist[i])
      downpart = downpart+(WeightValue[i]/constlist[i])

   x2 = x2up/downpart
   y2 = y2up/downpart

   F1 = 0
   F2 = 0
   for i in xrange(len(IDnodelib)):
      f1 = math.sqrt((x1-Xcoord[i])**2+(y1-Ycoord[i])**2+epsilon)
      F1 = F1+(WeightValue[i]*f1)
      f2 = math.sqrt((x2-Xcoord[i])**2+(y2-Ycoord[i])**2+epsilon)
      F2 = F2+(WeightValue[i]*f2)

   if abs(F2-F1)<=alfa:
      stop=1
   else:
      x1 = x2
      y1 = y2


IDsupernode = int(str(IDjunction)+str(0))

return IDsupernode, (x2,y2)
```

```
def new_coords(graf, coords):
  """
  Convert geographic coordinate system to orthogonal axes axis
  """
  Newgraf = nx.Graph()

  lon,lat, mlon, mlat = coords

  latitude = (lat+mlat)/2

  dely = 111132.954 - 559.822*(math.cos(2*latitude)) + 1.175*(math.cos(4*latitude))
  delx = math.pi/180*6378137*math.cos(math.atan(0.99664719*math.tan(latitude)))
  delx1 = math.pi/180*math.cos(latitude)*6367449
  print 'dely = ', dely
  print 'delx =', delx
  print 'delx1 =', delx1 #result with minus

  minx = 0
  miny = 0

  maxx = float(delx*mlon) - float(delx*lon)
  maxy = float(dely*mlat) - float(dely*lat)

  for node in graf.nodes():
    lon1, lat1 = graf.node[node]['pos']
    x1 = float(delx*lon1) - float(delx*lon)
    y1 = float(dely*lat1) - float(dely*lat)
    Newgraf.add_node(node, pos=(x1,y1))

  for edge1 in graf.edge:
    for edge2 in graf.edge[edge1]:
      Newgraf.add_edges_from([(edge1,edge2,graf.edge[edge1][edge2]), ])


  #newcoords_draw = [int(delx*lon), int(delx*mlon), int(dely*lat), int(dely*mlat)]
  newcoords_draw = [minx, maxx, miny, maxy]
  print 'newcoords_draw = ', newcoords_draw
  return Newgraf, newcoords_draw



######################################################################################
# 4 - functions for getting graphs with different configuration


def unite_two_grafs(graf1, graf2):
    """
    Fiction unites together two different graph.
    Avoidance of double writing for nodes and edges in a new graph.
    """

    graf = graf1
```

```
    grafedges = []
    for edge in graf.edges():
        grafedges.append(edge)

    grafnodes = []
    for node in graf.nodes():
        grafnodes.append(node)


    for edge in graf2.edges():
        edge1, edge2 = edge
        if grafedges.count(edge) == 0:
            graf.add_edges_from([(edge1,edge2,graf2.edge[edge1][edge2]), ])
        else:
            for attr in graf2[edge1][edge2]:
                graf[edge1][edge2][attr] = graf2[edge1][edge2][attr]

    for node in graf2.nodes():
        if grafnodes.count(node) == 0:
            graf.add_node(node, graf2.node[node])
        else:
            for attr in graf2.node[node]:
                graf.node[node][attr] = graf2.node[node][attr]

    return graf



def square_boundary(graf, coords):
    """
    Delete all nodes and edges which are situated outside of the square boundary
    """

    lon, lat, mlon, mlat = coords

    for node in graf.nodes():
      x, y = graf.node[node]['pos']
      if (lon <= x and x<= mlon) and (lat <= y and y <= mlat):
            pass
      else:
            graf.remove_node(node)

    return graf



def get_graf_one_key(graf,attr):
    """
    Create a graf, which includes only edges
    that have the same tag key-attribute (key-tag)
    Example:
    graf of all edges with key-attribute 'highway'
```

```
    """

    KeyGraf = nx.Graph() # maybe, directed graph is necessary

    for edge in graf.edges():
        edge1, edge2 = edge
        if graf.edge[edge1][edge2].has_key(attr):
            KeyGraf.add_edges_from([(edge1,edge2,graf.edge[edge1][edge2]), ])

    for node in KeyGraf.nodes():
        KeyGraf.node[node]["pos"] = graf.node[node]["pos"]

    return KeyGraf



def get_graf_one_key_clean(graf,attr):
    """
    This function does the same work as function "get_graf_one_key(graf,attr)",
    but even cleans graph from the repeated information
    #####Use it directly
    """

    KG = get_graf_one_key(graf,attr)
    ControlG = nx.Graph()

    for edge in KG.edges():
        edge1, edge2 = edge


        """CHECK UP"""
        # like after scanning:
        #print KG.edge[edge1][edge2]
        #e = 0
        #if KG.edge[edge1][edge2]['edge_numbers'] > 1:
        #    print
        #    print edge1, edge2
        #    print KG.edge[edge1][edge2]
        #    e =  1

        if ControlG.has_edge(edge1, edge2):
            print "We have problem in the function"
        else:
            ControlG.add_edges_from([(edge1,edge2,KG.edge[edge1][edge2]), ])


        if KG.edge[edge1][edge2]['edge_numbers'] == 1:
            #print KG.edge[edge1][edge2] #raw variation like after scanning
            if KG.edge[edge1][edge2].has_key('lib1'):
                del KG.edge[edge1][edge2]['lib1']
            #print KG.edge[edge1][edge2]

        else:
```

```
        numlist = []
        for j in range(KG.edge[edge1][edge2]['edge_numbers']):
          for i in KG.edge[edge1][edge2]['lib'+str(j+1)]:
            if i == attr:
              if numlist.count('lib'+str(j+1)) == 0:
                numlist.append('lib'+str(j+1))

        dellist = []
        for a in KG.edge[edge1][edge2]:
          if numlist.count(a) == 0 and a != 'edge_numbers':
            dellist.append(a)

        for a in dellist:
          del KG.edge[edge1][edge2][a]

        #print KG.edge[edge1][edge2]

        KG.edge[edge1][edge2]['edge_numbers'] = len(numlist)
        newnumlist = []
        for a in range(KG.edge[edge1][edge2]['edge_numbers']):
          newnumlist.append(('lib'+str(a+1)))

        #print KG.edge[edge1][edge2]

        for a in range(KG.edge[edge1][edge2]['edge_numbers']):
          temp = KG.edge[edge1][edge2][numlist[a]]
          del KG.edge[edge1][edge2][numlist[a]]
          KG.edge[edge1][edge2][newnumlist[a]] = temp
          KG.edge[edge1][edge2]['id'+str(a+1)] = KG.edge[edge1][edge2][newnumlist[a]]['id']
          for j in KG.edge[edge1][edge2][newnumlist[a]]:
            if KG.edge[edge1][edge2].has_key(j) == False:
              if j != 'id':
                KG.edge[edge1][edge2][j] = KG.edge[edge1][edge2][newnumlist[a]][j]

        if KG.edge[edge1][edge2]['edge_numbers'] == 1:
          del KG.edge[edge1][edge2]['lib1']

      """CHECK UP"""
      #if e == 1:
      #    print KG.edge[edge1][edge2]
      #if KG.edge[edge1][edge2]['edge_numbers'] > 1:
      #    print edge1, edge2
      #    print KG.edge[edge1][edge2]

      #ControlG.add_edges_from([(edge1,edge2,graf.edge[edge1][edge2]), ])


    return KG



def get_graf_one_value(graf, key, value):
    """
```

```
    Create a graf, which includes only edges
    that have the same tag pair of key-value
    Example:
    graf of all edges with tag pair key-value 'highway'-'motorway'
    """

    KeyGraf = get_graf_one_key(graf,key)
    ValueGraf = nx.Graph()

    for edge in KeyGraf.edges():
        edge1, edge2 = edge
        if KeyGraf[edge1][edge2][key] == value:
            ValueGraf.add_edges_from([(edge1,edge2,graf.edge[edge1][edge2]), ])

    for node in ValueGraf.nodes():
        ValueGraf.node[node]["pos"] = graf.node[node]["pos"]

    return ValueGraf



def get_graf_one_value_clean(graf, key, value):
    """
    Create a graf, which includes only edges
    that have the same tag pair of key-value
    Example:
    graf of all edges with tag pair key-value 'highway'-'motorway'
    This function does the same work as function "get_graf_one_value(graf,attr)",
    but is based on "get_graf_one_key_clean(graf,attr)"
    """

    KeyGraf = get_graf_one_key_clean(graf,key)
    ValueGraf = nx.Graph()

    for edge in KeyGraf.edges():
        edge1, edge2 = edge

        if KeyGraf[edge1][edge2]['edge_numbers'] == 1:
            if KeyGraf[edge1][edge2][key] == value:
                ValueGraf.add_edges_from([(edge1,edge2,KeyGraf.edge[edge1][edge2]), ])
                """CHECK UP"""
                #print edge1, edge2
                #print ValueGraf.edge[edge1][edge2]


        else:
            libs = []
            full = []
            empty = []
            for i in range(KeyGraf[edge1][edge2]['edge_numbers']):
                libs.append('lib'+str(i+1))
                if KeyGraf[edge1][edge2]['lib'+str(i+1)][key] == value:
                    full.append('lib'+str(i+1))
```

```
            else:
                empty.append('lib'+str(i+1))

        keyslib = []
        for j in KeyGraf[edge1][edge2]:
            if j != 'edge_numbers':
                keyslib.append(j)

        if len(libs) > len(empty):
            ValueGraf.add_edges_from([(edge1,edge2,KeyGraf.edge[edge1][edge2]), ])
            for j in keyslib:
                del ValueGraf[edge1][edge2][j]

            ValueGraf[edge1][edge2]['edge_numbers'] = len(full)
            write = 1
            for i in full:
                ValueGraf[edge1][edge2]['lib'+str(write)] = KeyGraf[edge1][edge2][i]
                ValueGraf[edge1][edge2]['id'+str(write)] = KeyGraf[edge1][edge2][i]['id']
                for k in KeyGraf[edge1][edge2][i]:
                    if k != 'id':
                        ValueGraf[edge1][edge2][k] = KeyGraf[edge1][edge2][i][k]

                write = write + 1

            if ValueGraf[edge1][edge2]['edge_numbers'] == 1:
                del ValueGraf[edge1][edge2]['lib1']

            """CHECK UP"""
            #print edge1, edge2
            #print 'origin'
            #print graf.edge[edge1][edge2]
            #print 'key'
            #print KeyGraf.edge[edge1][edge2]
            #print 'value'
            #print ValueGraf.edge[edge1][edge2]
            #print


        """CHECK UP"""
        #if ValueGraf.has_edge(edge1, edge2):
        #    print edge1, edge2
        #    print KeyGraf[edge1][edge2]
        #    print ValueGraf[edge1][edge2]


    for node in ValueGraf.nodes():
        ValueGraf.node[node]["pos"] = graf.node[node]["pos"]

    return ValueGraf



def create_road_graf(graf):
```

```
    """
    Get graf of roads: this graf includes all values of tag "highway"
    that are in the list of ROADATTRS
    """
    HighwayGraf = get_graf_one_key(graf,"highway")

    RoadsGraf = nx.Graph()

    for edge1 in HighwayGraf.edge:
     for edge2 in HighwayGraf.edge[edge1]:
         attr = [HighwayGraf.edge[edge1][edge2]['highway']]
         if is_road(attr):
          RoadsGraf.add_node(edge1, HighwayGraf.node[edge1])
          RoadsGraf.add_node(edge2, HighwayGraf.node[edge2])
          RoadsGraf.add_edges_from([(edge1,edge2, HighwayGraf.edge[edge1][edge2]), ])

    return RoadsGraf



def admin_boundary_graf(graf, level=None, cityname=None):
    """
    Create a graf of administrative boundaries of the cities, regions and so on
    """
    BGraf = get_graf_one_key(graf,'boundary')

    AdminGraf = nx.Graph()

    for edge1 in BGraf.edge:
     for edge2 in BGraf.edge[edge1]:
         attr = BGraf.edge[edge1][edge2]['boundary']
         #print attr
         if attr == 'administrative':
          AdminGraf.add_node(edge1, BGraf.node[edge1])
          AdminGraf.add_node(edge2, BGraf.node[edge2])
          AdminGraf.add_edges_from([(edge1,edge2, BGraf.edge[edge1][edge2]), ])


    controler = 0

    while controler == 0:
     finish = []

     for node in AdminGraf.nodes():
         if len(AdminGraf.neighbors(node)) <= 1:
          AdminGraf.remove_node(node)

     for node in AdminGraf.nodes():
         if len(AdminGraf.neighbors(node)) <= 1:
          finish.append(node)

     if len(finish) == 0:
         controler =1
```

```
# program proves a tag "admin_level"
if level != None:
  level_counter = []
  for edge1 in AdminGraf.edge:
    for edge2 in AdminGraf.edge[edge1]:
      if level_counter.count(AdminGraf.edge[edge1][edge2]['admin_level']) == 0:
        level_counter.append(AdminGraf.edge[edge1][edge2]['admin_level'])

  print level_counter
  use_level = level_counter[0]
  for l in level_counter:
    if int(use_level) > int(l):
      use_level = l

  print 'Administrative boundary level: ', use_level
  MainAdminGraf = nx.Graph()
  for edge1 in AdminGraf.edge:
    for edge2 in AdminGraf.edge[edge1]:
      if AdminGraf.edge[edge1][edge2]['admin_level'] == use_level:
        MainAdminGraf.add_edges_from([(edge1,edge2,AdminGraf.edge[edge1][edge2]), ])

  for node in MainAdminGraf.nodes():
    MainAdminGraf.node[node]["pos"] = AdminGraf.node[node]["pos"]

  controler = 0

     while controler == 0:
      finish = []

      for node in MainAdminGraf.nodes():
       if len(MainAdminGraf.neighbors(node)) <= 1:
        MainAdminGraf.remove_node(node)

      for node in MainAdminGraf.nodes():
       if len(MainAdminGraf.neighbors(node)) <= 1:
        finish.append(node)

      if len(finish) == 0:
       controler =1


# cityname - program proves a tag "name:left" or "name:right"
if cityname != None:
  NameAdminGraf = nx.Graph()
  for edge1 in AdminGraf.edge:
    for edge2 in AdminGraf.edge[edge1]:
      if AdminGraf.edge[edge1][edge2].has_key('name:left'):
        if AdminGraf.edge[edge1][edge2]['name:left'] == cityname:
          NameAdminGraf.add_edges_from([(edge1,edge2,AdminGraf.edge[edge1][edge2]), ])
      if AdminGraf.edge[edge1][edge2].has_key('name:right'):
        if AdminGraf.edge[edge1][edge2]['name:right'] == cityname:
          NameAdminGraf.add_edges_from([(edge1,edge2,AdminGraf.edge[edge1][edge2]), ])
```

```
    for node in NameAdminGraf.nodes():
      NameAdminGraf.node[node]["pos"] = AdminGraf.node[node]["pos"]

  if level != None:
    return MainAdminGraf
  elif cityname != None:
    return NameAdminGraf
  else:
    return AdminGraf



def road_graf_in_boundary(RoadsGraf, AdminGraf, GravitGraf, coords):
  """
  Creation a road graph which is covered only administrative territory
  """

  NewRG, newcoords_draw = new_coords(RoadsGraf, coords)
  NewAG, newcoords_draw = new_coords(AdminGraf, coords)
  NewGG, newcoords_draw = new_coords(GravitGraf, coords)

  nodelib = []
  for Gnode in NewGG.nodes():
    p1 = NewGG.node[Gnode]['pos']
    a1, a2 = p1#
    delnode = []
    for node in NewRG.nodes():
      p2 = NewRG.node[node]["pos"]
      b1, b2 = p2#
      for edge in NewAG.edges():
        i,j = edge
        p3 = NewAG.node[i]['pos']
        c1, c2 = p3#
        p4 = NewAG.node[j]['pos']
        d1, d2 = p4#
        a = Point(a1, a2)#
        b = Point(b1, b2)#
        c = Point(c1, c2)#
        d = Point(d1, d2)#
        if intersect(a,b,c,d) == True:#
        #if calculateIntersectPoint(p1, p2, p3, p4) != None:
          if delnode.count(node) == 0:
            delnode.append(node)

    for k in NewRG.nodes():
      if delnode.count(k) == 0:
        if nodelib.count(k) == 0:
          nodelib.append(k)


  for node in NewRG.nodes():
    if nodelib.count(node) == 0:
```

```
        NewRG.remove_node(node)

    for node in NewRG.nodes():
        NewRG.node[node]['pos'] = RoadsGraf.node[node]['pos']

    return NewRG



def is_road(attrs):
    """
    Return True if attributes correspond to a road
    """
    return has_attr(attrs,ROADATTRS)



def is_mainfeatures(attrs):
    """
    Return Treu if attributes correspond to a main features
    """
    return has_attr(attrs, MAINFEATURES)



def has_attr(attrs,attrs_comp):
    """
    Returns true if at least one attribute in attrs
    is in list attrs_comp
    Ex:
    attrs = ['author', 'highway']
    attrs_comp = ['highway','road']
    """
    ans = False
    for attr in attrs:
        ans = ans | (attr in attrs_comp)
    return ans



def get_graf_weighted_nodes(graf):
    """
    Return graf of all nodes (only nodes, without edges) that are weighted depending on a number
    of edges conected with each node
    With networkx-function "neighbors" must work easeir
    """
    WeightNodes = nx.Graph()

    for node in graf.nodes():
        k = 0
        for edge in graf.edge[node]:
            k = k+1
        WeightNodes.add_node(node, weight=k, pos=graf.node[node]['pos'])
```

```
    return WeightNodes


def get_graf_with_edges_lenghts(graf):
    """
    Return full graf with all weighted edges (weight = lenght, but name of weight is "lenght")
    """
    WeightEdges = nx.Graph()

    for edge1 in graf.edge:
     for edge2 in graf.edge[edge1]:
       lenght = get_len(graf, (edge1,edge2))
       WeightEdges.add_node(edge1, graf.node[edge1])
       WeightEdges.add_node(edge2, graf.node[edge2])
       WeightEdges.add_edges_from([(edge1,edge2, graf.edge[edge1][edge2]), ])
       WeightEdges.edge[edge1][edge2]['lenght'] = lenght

    return WeightEdges


def get_full_graf_weighted_nodes(graf):
    """
    Return full graf with all weighted nodes
    """
    WeightGraf = nx.Graph()

    WeightNodes = get_graf_weighted_nodes(graf)

    for edge1 in graf.edge:
     for edge2 in graf.edge[edge1]:
       WeightGraf.add_node(edge1, WeightNodes.node[edge1])
       WeightGraf.add_node(edge2, WeightNodes.node[edge2])
       WeightGraf.add_edges_from([(edge1,edge2, graf.edge[edge1][edge2]), ])

    return WeightGraf

##################################################################################
# 5 - functions for drawing grafs in different ways


def draw_graf(graf, coords=None):
    """
    Create a picture with help of grafs
    Coordinates does not still included!!!
    """
    if coords == None:
     coord_list = [13.72515, 13.73301, 51.0288, 51.0341]
    else:
     coord_list = []
```

```
    for i in coords:
        coord_list.append(i)

  plt.figure(1)

  print coord_list

  nx.draw_networkx_nodes(graf,pos=get_pos(graf),node_size=5)
  nx.draw_networkx_edges(graf,pos=get_pos(graf))

  plt.axis(coord_list)
  plt.show()



def draw_weighted_nodes(graf, WeightNodes, coords=None):
  """
  Graf picture where the weight of each node is showed
  WeightNodes is a graf that has all weighted nodes
  Coordinates does not still included!!!
  """

  if coords == None:
    coord_list = [13.72515, 13.73301, 51.0288, 51.0341]
  else:
    coord_list = []
    for i in coords:
        coord_list.append(i)

  ThreeWeightG = nx.Graph()
  FourWeightG = nx.Graph()
  MoreWeightG = nx.Graph()

  plt.figure(1)

  nx.draw_networkx_nodes(graf, pos=get_pos(graf), node_size=5)
  nx.draw_networkx_edges(graf, pos=get_pos(graf))

  for node in WeightNodes.nodes():
    if WeightNodes.node[node]['weight'] == 3:
      ThreeWeightG.add_node(node, WeightNodes.node[node])
    if WeightNodes.node[node]['weight'] == 4:
      FourWeightG.add_node(node, WeightNodes.node[node])
    if WeightNodes.node[node]['weight'] > 4:
      MoreWeightG.add_node(node, WeightNodes.node[node])

  if len(ThreeWeightG.nodes()) > 0:
    nx.draw_networkx_nodes(ThreeWeightG,      pos=get_pos(ThreeWeightG),      node_size=40,
node_color='yellow')
  if len(FourWeightG.nodes()) > 0:
    nx.draw_networkx_nodes(FourWeightG,      pos=get_pos(FourWeightG),      node_size=60,
node_color='green')
  if len(MoreWeightG.nodes()) > 0:
```

```
    nx.draw_networkx_nodes(MoreWeightG, pos=get_pos(MoreWeightG), node_size=20,
node_color='blue')

  plt.axis(coord_list)
  plt.show()




def draw_grafs_as_layers(RoadsGraf, AdminGraf, SupernodeGraf=None, G=None, T=None, H=None,
coords=None):# SupernodeGraf
  """
  Draw different grafs as layers: one graf - one layer
  """

  if coords == None:
   coord_list = [13.72515, 13.73301, 51.0288, 51.0341]
  else:
   coord_list = []
   for i in coords:
       coord_list.append(i)


  plt.figure(1)

  #1 - Layer Roads

  if len(RoadsGraf.nodes()) > 0:
    #nx.draw_networkx_nodes(RoadsGraf, pos=get_pos(RoadsGraf), node_size=5)
    nx.draw_networkx_edges(RoadsGraf, pos=get_pos(RoadsGraf), width=1.0, edge_color='green')

  #2 - Layer Administrative Boundaries

  if len(AdminGraf.nodes()) > 0:
    nx.draw_networkx_nodes(AdminGraf, pos=get_pos(AdminGraf), node_size=5)
    nx.draw_networkx_edges(AdminGraf, pos=get_pos(AdminGraf), width=4.0, edge_color='red')

  #3 - Layer Supernode
  if SupernodeGraf != None:
    if len(SupernodeGraf.nodes()) > 0:
      nx.draw_networkx_nodes(SupernodeGraf, pos=get_pos(SupernodeGraf), node_size=50,
node_color='blue')
      #nx.draw_networkx_edges(SupernodeGraf, pos=get_pos(SupernodeGraf), width=1.0,
edge_color='blue')

  #4
  if G != None:
    if len(G.nodes()) > 0:
      nx.draw_networkx_nodes(G, pos=get_pos(G), node_size=50, node_color='yellow')
      #nx.draw_networkx_edges(G, pos=get_pos(G), width=1.0, edge_color='yellow')

  #5
  if T != None:
```

```
        if len(T.nodes()) > 0:
            nx.draw_networkx_nodes(T, pos=get_pos(T), node_size=50, node_color='orange')
            nx.draw_networkx_edges(T, pos=get_pos(T), width=1.0, edge_color='orange')

    #6
    if H != None:
        if len(H.nodes()) > 0:
            #nx.draw_networkx_nodes(H, pos=get_pos(H), node_size=50, node_color='brown')
            nx.draw_networkx_edges(H, pos=get_pos(H), width=1.0, edge_color='brown')

    plt.axis(coord_list)
    plt.show()



def draw_road_types(AdminGraf, MotorwayFull=None, TrunkFull=None, PrimaryFull=None, SecondaryFull=None, TertiaryFull=None, ResidGraf=None, LivingStrGraf=None, RoadGraf=None, TrackGraf=None, ServiceGraf=None, UnclassGraf=None, CyclewayGraf=None, CyclelementsGraf=None, BiciGraf=None, coords=None):
    """
    Draw different grafs as layers: one graf - one road type - one layer
    """

    if coords == None:
      coord_list = [13.72515, 13.73301, 51.0288, 51.0341]
    else:
      coord_list = []
      for i in coords:
            coord_list.append(i)


    plt.figure(1)

    #1 - Layer Administrative Boundaries

    if len(AdminGraf.nodes()) > 0:
        nx.draw_networkx_nodes(AdminGraf, pos=get_pos(AdminGraf), node_size=5)
        nx.draw_networkx_edges(AdminGraf, pos=get_pos(AdminGraf), width=4.0, edge_color='red')

    #2 - Layer Motorways and Motorway Links
    if MotorwayFull != None:
        if len(MotorwayFull.nodes()) > 0:
            nx.draw_networkx_edges(MotorwayFull, pos=get_pos(MotorwayFull), width=1.0, edge_color='DarkViolet')

    #3 - Layer Trunks and Trunk Links
    if TrunkFull != None:
        if len(TrunkFull.nodes()) > 0:
            nx.draw_networkx_edges(TrunkFull, pos=get_pos(TrunkFull), width=1.0, edge_color='blue')

    #4 - Layer Primaries and Primary Links
    if PrimaryFull != None:
        if len(PrimaryFull.nodes()) > 0:
```

```
        nx.draw_networkx_edges(PrimaryFull,              pos=get_pos(PrimaryFull),              width=1.0,
edge_color='DeepSkyBlue')

    #5 - Layer Secondaries and Secondary Links
    if SecondaryFull != None:
      if len(SecondaryFull.nodes()) > 0:
        nx.draw_networkx_edges(SecondaryFull,            pos=get_pos(SecondaryFull),            width=1.0,
edge_color='ForestGreen')

    #6 - Layer Tertiaries and Tertiary Links
    if TertiaryFull != None:
      if len(TertiaryFull.nodes()) > 0:
        nx.draw_networkx_edges(TertiaryFull,             pos=get_pos(TertiaryFull),             width=1.0,
edge_color='YellowGreen')

    #7 - Layer Residential Roads
    if ResidGraf != None:
      if len(ResidGraf.nodes()) > 0:
        nx.draw_networkx_edges(ResidGraf,                pos=get_pos(ResidGraf),                width=1.0,
edge_color='PaleGreen')

    #8 - Layer Living Streets
    if LivingStrGraf != None:
      if len(LivingStrGraf.nodes()) > 0:
        nx.draw_networkx_edges(LivingStrGraf,            pos=get_pos(LivingStrGraf),            width=1.0,
edge_color='yellow')

    #9 - Layer Roads
    if RoadGraf != None:
      if len(RoadGraf.nodes()) > 0:
        nx.draw_networkx_edges(RoadGraf, pos=get_pos(RoadGraf), width=1.0, edge_color='orange')

    #10 - Layer Tracks
    if TrackGraf != None:
      if len(TrackGraf.nodes()) > 0:
        nx.draw_networkx_edges(TrackGraf,                pos=get_pos(TrackGraf),                width=1.0,
edge_color='HotPink')

    #11 - Layer Service
    if ServiceGraf != None:
      if len(ServiceGraf.nodes()) > 0:
        nx.draw_networkx_edges(ServiceGraf,              pos=get_pos(ServiceGraf),              width=1.0,
edge_color='crimson')

    #12 - Layer Unclassified roads
    if UnclassGraf != None:
      if len(UnclassGraf.nodes()) > 0:
        nx.draw_networkx_edges(UnclassGraf,              pos=get_pos(UnclassGraf),              width=1.0,
edge_color='brown')

    #13 - Layer Cycleways
    if CyclewayGraf != None:
      if len(CyclewayGraf.nodes()) > 0:
```

```
        nx.draw_networkx_edges(CyclewayGraf,         pos=get_pos(CyclewayGraf),         width=2.0,
edge_color='black')

    #14 - Layer Bicycle infrastructure on other road types
    if CyclelementsGraf != None:
        if len(CyclelementsGraf.nodes()) > 0:
            nx.draw_networkx_edges(CyclelementsGraf,     pos=get_pos(CyclelementsGraf),     width=2.0,
edge_color='black')

    #15 - Layer Bicycle Allowed
    if BiciGraf != None:
        if len(BiciGraf.nodes()) > 0:
            nx.draw_networkx_edges(BiciGraf, pos=get_pos(BiciGraf), width=2.0, edge_color='grey')

    plt.axis(coord_list)
    plt.show()


##########################################################################################

# 6 - functions for parameters calculation


def get_length_key_value(graf, key, value, mainkey=None, mainvalue=None):
    """
    Calculation of total road lenght
    Type of road is set by key and value (possible to use two levels of key-value filter)
    """

    main = 1
    if mainkey == None:
        mainkey = key
        main = 0
    if mainvalue == None:
        mainvalue = value
        main = 0

    #G = get_graf_one_value_clean(graf, mainkey, mainvalue)

    if main == 1:
        K = get_graf_one_value_clean(graf, mainkey, mainvalue)
        G = get_graf_one_value_clean(K, key, value)
    else:
        G = get_graf_one_value_clean(graf, mainkey, mainvalue)

    print mainkey
    print mainvalue
    print key
    print value

    length = 0

    for edge in G.edges():
```

```
        edge1, edge2 = edge
        l = get_len(G,(edge1,edge2))

        length = length + l

    print length
    return length#, G



def return_list_of_keys(graf, key):
    """
    Creation of value list according the given key
    """

    values_list = []
    for edge in graf.edges():
        edge1, edge2 = edge
        if graf.edge[edge1][edge2].has_key(key) == True:
            if graf.edge[edge1][edge2]['edge_numbers'] == 1:
                if values_list.count(graf.edge[edge1][edge2][key]) == 0:
                    values_list.append(graf.edge[edge1][edge2][key])
            else:
                for i in range(graf.edge[edge1][edge2]['edge_numbers']):
                    if graf.edge[edge1][edge2]['lib'+str(i+1)].has_key(key) == True:
                        if values_list.count(graf.edge[edge1][edge2]['lib'+str(i+1)][key]) == 0:
                            values_list.append(graf.edge[edge1][edge2]['lib'+str(i+1)][key])

    return values_list



##########################################################################################
#
```

*names of functions are marked with red color and bold type

*names of sections are marked with bold type