COVER PAGE

DELIVERABLE

Project Acronym: i-locate

Grant Agreement number: 621040

Project Title: Indoor/outdoor LOCation and Asset management Through open gEodata

D3.3 – Routing Service
(accompanying report)

Revision: v.1.3

Authors:

Tao Feng (TUE)

Theo Arentze (TUE)

Joran Jessurun (TUE)

Conti Giuseppe (TRILOGIS)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisat	Description
v.0.1	2015/05/18	Tao Feng	TUE	Main structure
v.0.2	2015/05/21	Theo Arentze	TUE	Revised structure
v.0.3	2015/06/10	Tao Feng	TUE	Added contents
v.0.4	2015/06/11	Theo Arentze	TUE	Revised contents
v.0.5	2015/06/16	Tao Feng	TUE	Revision
v.0.6	2015/06/17	Theo Arentze	TUE	Revision
v.0.7	2015/06/18	Tao Feng	TUE	Revision
v.0.8	2015/06/18	Joran Jessurun	TUE	Added and revised indoor graph builder and indoor geocoder
v.0.9	2015/06/21	Giuseppe Conti	TRILOGIS	Complete review
v.1.0	2015/06/21	Theo Arentze	TUE	Complete review
v.1.1	2015/06/23	Joran Jessurun	TUE	Revised indoorGML and figures
v.1.2	2015/06/23	Tao Feng	TUE	Revision
v.1.3	2015/06/25	Giuseppe Conti	TRILOGIS	Complete review

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

1 List of references

Number	Full reference
1	i-locate deliverable D1.1 – Use cases description and Privacy Threat Vulnerability and Risk Analysis. Available online from: http://www.i-locate.eu/public-deliverables/
2	i-locate deliverable D.1.4 – System Architecture. Available online from: http://www.i-locate.eu/public-deliverables/
3	i-locate deliverable D.1.4 – System Architecture Appendix-1. Available online from: http://www.i-locate.eu/public-deliverables/
4	i-locate deliverable D.1.4 – System Architecture Appendix-2. Available online from: http://www.i-locate.eu/public-deliverables/
5	i-Locate deliverable D.3.1 (Accompanying report) (update 1) – i-locate Toolkit v1.0 – Available online from: http://www.i-locate.eu/public-deliverables/
6	i-Locate deliverable D.3.1 – i-locate Toolkit v1.0 (Prototype) – i-locate Toolkit v1.0 – Available online from: http://www.i-locate.eu/public-deliverables/
7	i-Locate deliverable D.3.2 – Localisation and Tracking System (Prototype) – Available online from: http://www.i-locate.eu/public-deliverables/
8	IndoorGML – Available online from: http://indoorgml.net/
9	iTour, Intelligent Transport system for Optimized URban trips (i-Tour) –Available online from: http://www.transport-research.info/web/projects/project_details.cfm?id=41605
10	OTP, OpenTripPlanner – Available online from: http://www.opentripplanner.org/
11	OSM, OpenStreetMap - Available online from: https://www.openstreetmap.org/
12	GTFS, General Transit Feed Specification – Available online from: https://developers.google.com/transit/gtfs/

2 Table of Acroyms

Acronym	Descriptiobn
Anchor node	Special type of nodes connecting graphs
API	Application Program Interface, a set of routines, protocols, and tools for building software applications.
CSV	Comma-separated values (*CSV*)
GTFS	General Transit Feed Specification, a common format for public transportation schedules and associated geographic information
IndoorGML	A candidate OGC standard for an open data model and XML schema for indoor spatial information
JSON	JavaScript Object Notation, an open standard format that uses human-readable text to transmit data objects
LBS	Location Based Services, services offered through a mobile phone and take into account the device's geographical location
MLSM	Multi-Layered Space Model
OSM	OpenStreetMap, an openly licensed map of the world being created by volunteers using local knowledge, GPS tracks and donated sources.
OTP	OpenTripPlanner, an open source multi-modal trip planner.
XML	Extensible Markup Language (*XML*), a markup language that defines a set of rules for encoding documents

3 Executive Abstract

This deliverable reports the development of the routing system for indoor and outdoor navigation. The routing system is a part of the middleware of the integrated system for indoor/outdoor location and asset management through open geodata (i-Locate).

The work described in this deliverable is an accompanying report of a running system. It was developed based on the OpenTripPlanner (OTP) software, which is an open-source multimodal routing system. Where the OTP is designed specifically for outdoor multimodal routing, we extend OTP with additional indoor routing functionality and integration of indoor and outdoor routing.

The integrated indoor and outdoor routing system runs as a service and works seamlessly with the i-Locate toolkit and i-Locate portal. It automatically imports the indoor graphs which are created using the portal and builds an integrated navigation graph together with the outdoor road network and public transport data. When a request from the client in the toolkit is received, the routing system generates the optimal route along with the turn-by-turn navigation information as a response to provide guidance for navigation.

The routing system extends the OTP by additionally developing the indoor graph builders, indoor routing algorithms and indoor turn-by-turn directions. The developed graph builder supports the IndoorGML format, which is the current candidate standard for indoor mapping. Different graphs (sub-graphs at different floors and/or outdoor graphs) are interconnected using the pre-defined anchor nodes. The optimal route is based on the general costs where different weight factors are considered in terms of the type of indoor edges, such as stairs and elevators. Special emphasis is also put to the case of using wheelchairs in the sense that the stairs should be avoided. Moreover, the turn-by-turn directions for indoor navigation are specifically developed. The relative directions are consistent with the outdoor navigation directions with special focus on the use of stairs and elevators.

It should be noted that the routing system developed in the i-Locate project is completely based on the open-source standard and open data standards, which benefits at the maximum to the public. The routing request and response formats are easily to be adapted to comply with Open Location Services Interface Standard (OpenLS).

4 Table of content

1 LIST OF REFERENCES 3
2 TABLE OF ACROYMS 4
3 EXECUTIVE ABSTRACT 5
4 TABLE OF CONTENT 6
5 TABLE OF FIGURES 7
6 TABLE OF TABLES 8
7 INTRODUCTION 9
1.1. Scope 9
1.2. Routing service - requirements 9
1.2.1. Indoor routing 9
1.2.2. Outdoor routing 10
1.2.3. Integrated indoor-outdoor routing 10
8 NAVIGATION GRAPH 12
1.3. Indoor navigation graph 12
1.3.1. The indoor data model 12
1.3.2. IndoorGML 13
1.4. Outdoor navigation graph 14
1.4.1. OSM 15
1.4.2. GTFS 15
1.5. Graph builder 16
1.5.1. Indoor graph builder 16
1.5.2. Outdoor graph builder 19
9 GENERATING ROUTES 21
1.6. Indoor routing 21
1.6.1. Route planning 21
1.6.2. Turn-by-turn directions 22
1.7. Outdoor routing 23
1.7.1. Route planning 23
1.7.2. Turn-by-turn directions 24
1.8. Routing-system APIs 24
1.8.1. Routing requests 24
1.8.2. Routing responses 26
10 INDOOR GEOCODER 30
1.9. Geocoder request 30
1.10. Geocoder response 30
11 CONCLUSIONS 32

5 Table of Figures

Fig. 1: Method of building an indoor graph 19

Fig. 2: Screenshot of the indoor and outdoor routing service 27

6 Table of Tables

Table 1: Indoor data model 12

Table 2: Data files and required fields in GTFS feed 15

Table 3: Graph builder modules in OTP for outdoor routing 16

Table 4: Extended graph builder modules for indoor routing 17

Table 5: Turn-by-turn direction for indoor navigation 23

Table 6: Travel modes in OTP 23

Table 7: Turn-by-turn direction for outdoor navigation 24

Table 8: Routing request (extended from OTP) 25

Table 9: Parameters for optimized route 25

Table 10: Routing response parameters (extended from OTP) 26

Table 11: Parameters of geocoder request 30

Table 12: Parameters of geocoder response 31

7 Introduction

1.1. Scope

Routing services play a fundamental role in the development of navigation systems. With given start and end locations, these services calculate the optimal route. Whereas outdoor route planning and navigation has been well developed, the development of indoor navigation has been delayed for long because of the different limitations in techniques for indoor localization. The i-Locate project uses localization techniques in combination with a multimodal routing system which have reached maturity in recent years (as demonsrated within i-Locate deliverables).

This document describes the routing service that is developed and implemented in the context of Task 3.3 of the i-Locate project for integrated outdoor and indoor route planning and navigation. As the base technology, OpenTripPlanner (OTP, http://www.opentripplanner.org/), an open source platform for outdoor multi-modal routing, is used. Originally, the i-Tour route planner (http://www.transport-research.info/web/projects/project_details.cfm?id=41605) was proposed as the base technology for this task.

During the project, however, it was decided to use OpenTripPlanner as the base technology. The main reasons for the change of platform were that OTP has been optimized in recent years to offer the same functionality as i-Tour, with the advantage that it requires less memory capacity and computation time for graph building due to a more efficient transport network modeling technique (a time dependent instead of a time expanded approach).

The new routing service works for both indoor and outdoor routing, with specific added value to indoor routing and indoor graph building. The routing service supports multimodal routing from door-to-door in a seamless way across outdoor and indoor environments. It allows the navigation between multiple floors by incorporating stairs, elevators, open spaces, etc., and generates turn-by-turn navigation guidance which is specific for indoor routing.

1.2. Routing service - requirements

1.2.1. Indoor routing

As said, the routing service integrates outdoor and indoor routing. The purpose of indoor routing is to provide an optimal route for navigation when the user is inside the building. Indoor routing involves unique features which are different from outdoor routing. Unlike the outdoor case, it is common to travel between different floors in indoor environment. This brings along the issue of how to connect the routing graphs between different floors. The floors are connected by stairs, elevators, etc. Therefore, a navigation graph which incorporates the elevators and stairs should be built.

With the navigation graph, the routing algorithm calculates the optimal route depending on the transport mode. As indoor trips involve walking or wheelchair as possible transport modes, the optimal route should represent the specifics of these two modes, whereas the main transport mode for indoor trips is walking.

For the indoor and outdoor case irrespectively, the optimal route is the route that minimizes travel costs (resistance). By default the routing service assumes that travel costs is determined by travel time so that the least-cost route is equal to the fastest route. However, specific resistance factors should be taken into

account. For instance, the travel time by walking is different for walking between general corridors and stairs or elevators. In addition, the provided route should also allow the choice of wheelchair. This means the routing should consider whether a link in the graph is traversable for wheelchairs.

Apart from that, the routing system provides turn-by-turn indoor navigation directions for the optimal route. The turn-by-turn information provides the guidance regarding directions and steps of the route.

As for indoor data, the development of a data standard is still on-going. Discussions on the indoor data format have been increasingly taken place over recent years to define a proper data format for navigation. There has been open discussions in the OpenStreetMap forums to address the issue of an indoor data standard. However, a consensus is still far to reach. In i-Locate, the IndoorGML has been selected as the standard for indoor data. This standard provides solutions for rich graph representations of indoor environments and serves the purpose of not only routing but also representation of indoor environments in 2D and 3D contexts. Thus, the i-Locate routing service we developed supports the IndoorGML data format.

1.2.2. Outdoor routing

Outdoor routing refers to the service of providing an optimal route for the outdoor part of the trip. Existing trip planners are mainly designed for uni-modal routing in the sense that the optimal route is given based on a specific single transportation mode, e.g. car or public transport. The option of using joint travel modes is not available in the uni-modal trip planners. In order to offer more travel options in reality, recent advancement has been introduced to the development of multimodal route planners where the combined use of multiple travel modes is supported. This generally involves the support of constructing an integrated network (a supernetwork) based on data of both the road network and public transport networks. The routing algorithm based on the transportation infrastructure data and public transport data then can provide personalized routing service for individuals.

The OpenTripPlanner (OTP) is such a full-fledged open-source multimodal routing system (http://www.opentripplanner.org/). It supports the combined use of multiple travel modes, i.e. routes through a multimodal network that involve a park and ride (P+R) or bike and ride (B+R) facility. The routing service for outdoor multimodal routing is provided based on OpenStreetMap data (road networks) and General Transit Feed Specification (GTFS) data (public transport), both of them are open data. In the development of the i-Locate routing system, we have adopted OpenTripPlanner (OTP) as the underlying technology for software development.

1.2.3. Integrated indoor-outdoor routing

It is important to provide users the route which offers cohesive guidance information for trips to and from indoor locations. Therefore, the i-Locate routing service supports both indoor and outdoor routing, with a specific definition of the indoor data structure. The user specifies the start and end locations in terms of longitude, latitude and floor level co-ordinates. Any start and end combination of indoor and outdoor location is supported, i.e., indoor-indoor, outdoor-outdoor, outdoor-indoor and indoor-outdoor.

The service gives the optimal route plan with complete turn-by-turn directions which can guide people while they travel between indoor and outdoor environments. The routing service supports multimodal routing from door-to-door in a seamless way across outdoor and indoor environments. It allows the navigation among multiple floors where traversing through stairs, elevators, open spaces, etc. are

implemented. The integrated routing service uses OTP for outdoor routing and extends the OTP to support also indoor routing.

In summary, the i-Locate routing system is an extension of OTP. It has the following functional and technical features (extended from OTP):

- Supports integrated outdoor-indoor routing.
- Supports multi-modal routing.
- Uses the time-dependent approach to represent schedules of public transport.
- Uses a hierarchical network representation to increase computation efficiency.
- Plans multi-modal walking, wheelchair, bicycle and transit trips.
- Takes travel time, road type, safety, and elevation into account, and allows users to customize the weighting of these three factors.
- Shows graphical elevation profiles for bike trips.
- Imports data from GTFS, OpenStreetMap, digital elevation models and IndoorGML data.
- Typically provides multiple itineraries in a fraction of a second even in large metropolitan networks.
- Exposes a web services API which other apps or front-ends can build on.
- Supports GTFS-Realtime for service changes and alerts in both polling (pull) and streaming (push) modes.
- Supports bike rental with dynamic availability information, as well as park-and-ride and kiss-and-ride.
- Supports one-to-many and many-to-many searches for planning, alternatives analysis, and accessibility research purposes.

8 Navigation Graph

The navigation graph includes the information required for routing algorithms. Similar to a road network, a navigation graph represents the topological connection between edges and vertices. In order to serve the cohesive routing for outdoor and indoor environment, an integrated navigation graph which incorporates both indoor and outdoor navigation graphs is a premise. For the outdoor environment, the i-Locate routing system uses OSM and GTFS data to present the road and public transport data, respectively. For the indoor environment, the navigation graph is created through the i-Locate portal. In this section, we define the basic elements of indoor graphs required for routing and assumed by the i-Locate routing system.

The graph of the building (or multiple buildings) needs to be connected with the outdoor road and public-transport network. There is therefore the issue of building the integrated navigation graph. The integrated navigation graph is built by connecting the outdoor and indoor networks through pre-defined so-called anchor nodes (building entrances). A proper indoor data model to connect indoor graphs with outdoor graphs is developed. In that data model, the anchor node is defined as an attribute of the indoor vertex, which is used to connect multiple indoor/outdoor graphs as a single integrated graph. The combined outdoor-indoor navigation graph is then used by the routing algorithm to find the optimal route.

1.3. Indoor navigation graph

1.3.1. The indoor data model

The indoor data model defines all needed elements for creating the navigation graph. It also represents the property for each element needed in an indoor navigation graph. The model defined here is generic and independent of the specific data format used. *Node* and *way* are the basic entities in the data model (Table 1).

Table 1: Indoor data model

Nodes	• Longitude and latitude, (double) • Level: the level where the node is located (double) • Name: the name of the node (String) • Ref: reference information of the node, e.g. doctor name, functionality etc. (String) • Is Anchor: is the node is an anchor node (true or false)

Ways	• Name: name of the way (String)
	• Type:
	  ○ footway
	  ○ stair
	  ○ ramp
	  ○ elevator
	  ○ escalator
	  ○ service
	• Wheelchair accessible (true or false)
	• Ref: reference information (String)

A graph is composed by nodes and ways. The nodes represent connections between ways, such as cross sections of corridors, doors and the connecting point between floors. All nodes for indoor environments should have the property of level which represents the floor number. It is assumed that all nodes without a level associated to it have the value of level 0, i.e. the default is level = 0. The level is a double value, which allows the representation of cases such as a half-way level between two main floors, or a room at the platform connecting stairs.

If the node represents a door the name represents the name of the room. It is also used for the indoor geocoder built into the i-Locate routing system. The name of rooms should be specific for the building to avoid ambiguity when rooms of different buildings would have the same name.

Anchor nodes have two possible roles: the first one, is to connect the indoor graph with the outdoor graph, the second is to connect multiple floors. The various ways represent the connecting lines between different locations, like corridors, stairs and elevators. The level attached to the start and end nodes of the way represent the level of the way. Note that the levels of start and end nodes do not have to be equal. For example, the levels of the start and end nodes of a stair can be 1 and 2.

A property of whether the link is wheelchair accessible is attached to the way. This information will be considered in the routing algorithm where ways that are not wheelchair accessible are avoided in case of using wheelchair.

1.3.2. IndoorGML

The IndoorGML format is used to describe the indoor graphs in i-Locate. IndoorGML has been developed by OGC with a target to be the standard of indoor spatial information (indoorGML, http://indoorgml.net/). IndoorGML has two conceptual frameworks namely Structured Space Model and Multi-Layered Space Model (MLSM). For details regarding the models refer to the IndoorGML website and additional documentation (http://indoorgml.net/).

IndoorGML has comprehensive solutions regarding data organization, such as the rooms, corridors, stairs and elevators for indoor data. IndoorGML defines the anchor node as additional topology to connect indoor and outdoor space. The anchor node differs from other nodes in the topological graph. For example, an entrance of a building can be treated as an anchor node since it can be used to connect indoor and outdoor spaces.

The newest version (Version 1) is based on the requirements from indoor navigation due to standardization demands, such as indoor LBS, routing services, and emergency control in indoor space. This suits well our purpose to provide a proper routing service.

The approach we developed here is mostly consistent with the current definitions of IndoorGML standard. For i-Locate the choice has been made to not use Cellular space and external references to include semantic information needed for the indoor navigation graph. Instead extra attributes and elements have been added to SpaceLayers, States and Transition elements.

The SpaceLayer has an extra attribute that indicates what kind of navigation graph is being represented. It can be a walk, wheelchair or another kind of navigation graph. State elements (nodes / vertices) add the attributes isDoor and isAnchor to indicate if a node is an anchor node and / or a door node. Transitions include the transiationType element to indicate if an edge is a normal edge, an elevator edge, a stair edge or another kind of edge.

Below is an excerpt of the parts of the IndoorGML XMLSchema that have been changed for this.

```
<xs:complexType name=" SpaceLayerType" >
    <xs:complexContent>
        <xs:extension base=" gml:AbstractFeatureType" >
            <xs:sequence>
                <xs:element name=" usage"  type=" gml:CodeType"  minOccurs=" 0"  maxOccurs=" unbounded" />
                <xs:element name=" terminationDate"  type=" xs:dateTime"  minOccurs=" 0"  maxOccurs=" 1" />
                <xs:element name=" function"  type=" gml:CodeType"  minOccurs=" 0"  maxOccurs=" unbounded" />
                <xs:element name=" creationDate"  type=" xs:dateTime"  minOccurs=" 0"  maxOccurs=" 1" />
                <xs:element name=" class"  type=" SpaceLayerClassTypeType"  minOccurs=" 0"  maxOccurs=" 1" />
                <xs:element name=" nodes"  type=" NodesType"  minOccurs=" 1"  maxOccurs=" unbounded" />
                <xs:element name=" edges"  type=" EdgesType"  minOccurs=" 0"  maxOccurs=" unbounded" />
                <xs:element name=" navigationType"
                        type=" typeOfNavigationGraphEnumerationType"  minOccurs=" 0"  maxOccurs=" 1" /> <!--
                 added -->
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:simpleType name=" typeOfNavigationGraphEnumerationType" >
    <xs:restriction base=" xs:string" >
        <xs:enumeration value=" WALK" />
        <xs:enumeration value=" WHEELCHAIR" />
        <xs:enumeration value=" OTHER" />
    </xs:restriction>
</xs:simpleType>

<xs:complexType name=" StateType" >
    <xs:complexContent>
        <xs:extension base=" gml:AbstractFeatureType" >
            <xs:sequence>
                <xs:element name=" duality"  type=" CellSpacePropertyType"  minOccurs=" 0"  maxOccurs=" 1"  />
                <xs:element name=" connects"  type=" TransitionPropertyType"  minOccurs=" 0"  maxOccurs="
unbounded"  />
                <xs:element name=" geometry"  type=" gml:PointPropertyType"  minOccurs=" 0"  maxOccurs=" 1"  />
            </xs:sequence>
            <xs:attributeGroup ref=" gml:AssociationAttributeGroup"  />
            <xs:attribute name=" isAnchorNode"  type=" xs:boolean"  default=" false"  /> <!-- added -->
            <xs:attribute name=" isDoor"  type=" xs:boolean"  default=" false" /> <!-- added -->
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name=" TransitionType" >
    <xs:complexContent>
        <xs:extension base=" gml:AbstractFeatureType" >
            <xs:sequence>
                <xs:element name=" weight"  type=" xs:double"  minOccurs=" 0"  maxOccurs=" 1" />
                <xs:element name=" start"  type=" StatePropertyType" />
                <xs:element name=" end"  type=" StatePropertyType" />
                <xs:element name=" duality"  type=" CellSpaceBoundaryPropertyType"  minOccurs=" 0"  maxOccurs=" 1"
/>
                <xs:element name=" geometry"  type=" gml:CurvePropertyType"  minOccurs=" 0"  maxOccurs=" 1" />
                <xs:element name=" transitionType"  type=" typeOfTransitionEnumerationType"  minOccurs=" 0"
                        maxOccurs=" 1"  /> <!-- added -->
            </xs:sequence>
            <xs:attributeGroup ref=" gml:AssociationAttributeGroup" />
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:simpleType name=" typeOfTransitionEnumerationType" >
    <xs:restriction base=" xs:string" >
        <xs:enumeration value=" NORMAL" />
        <xs:enumeration value=" ELEVATOR" />
        <xs:enumeration value=" STAIRS" />
        <xs:enumeration value=" OTHER" />
    </xs:restriction>
</xs:simpleType>
```

1.4. Outdoor navigation graph

The outdoor navigation graph represents the essential elements for outdoor routing service. It includes the transportation infrastructure related to road network and public transport data. For the road network regarding the use of private transportation modes, i-Locate uses OpenStreetMap data. For public transport, it uses GTFS data.

1.4.1. OSM

OSM includes the data of roads, buildings, land uses and points of interest, which serves various applications including navigation services. To build the navigation graph, the i-Locate routing system uses OSM data of the road network for car, bike and pedestrian. There are also detailed properties for each type of road.

The basic component of the OSM data model is called Element. An element can be a node, a way or a relation. A node is a single point in space defined by latitude, longitude and node id. A way represents a main road, a railway line or some other stream. A way is basically an ordered list of nodes. The properties of a way are represented using one or more tags. A way can be open or closed. A closed way is one whose last node on the way is also the first on that way. A closed way may be interpreted either as a closed polyline, an area, or both. A relation is used to define logical or geographic relationships between elements. It consists of tags and an ordered list of nodes, ways and/or relations. For the routing purpose specifically, the speed information, length information and type of the road (i.e. highway, local road) is used to calculate optimal routes (fastest or shortest route).

1.4.2. GTFS

To plan public transport trips, data of schedules (time tables) and routes of public transport services are needed. The i-Locate routing system (OTP) uses the General Transit Feed Specification (GTFS) format for public transport schedules and associated route information. A GTFS feed is composed of a series of text files (e.g., collected in a ZIP file). Each file models a particular aspect of transit information: stops, routes, trips, and other schedule data. Transit agencies can share their public transit information through GTFS feed which can be used by developers in various applications, i.e. trip planners. A detailed description of the GTFS data format can be found online from https://developers.google.com/transit/gtfs.

Specifically, GTFS data for a public transport system is a collection of Comma Separated Values (CSV) files. Each file or table describes a part of operations of the transit system. Each file has the name which represents the contents of the table. The list of required files is shown in Table 2.

The agency table provides information about the transit agency, including name, website and contact information. The routes table defines distinct routes of public transport services using the route id, name and type of the route as descriptors. The trip table represents the unique trips defined by a trip id and the relations between the trip and the route and service respectively using route id and service id. The stops table defines the geographic locations of stops or stations in the transit system. It has the detailed information regarding arrival time, departure time and stop sequence for each stop of the trip. The calendar table defines service patterns that operate recurrently, i.e. every weekday. In addition, there are also some optional data files, including the calendar_dates.txt, fare_attributes.txt, fare_rules.txt, shapes.txt, frequencies.txt, transfers.txt and feed_info.txt.

Table 2: Data files and required fields in GTFS feed

File	Required fields
agency	Name, url, time zone
route	route_id, short name, long name, type
trips	trip_id, route_id, service_id
stop_times	stop_id, trip_id, arrival_time, departure_time, stop_sequence

stops	stop_id, stop_name, stop_lon, stop_lat
calendar	service_id, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, start_date, end_date

The i-Locate routing system uses GTFS data to build the integrated graph for routing of public transport trips using the OTP graph builder. The geo-locations of transit stops and stations are used to connect the transit data with road networks. The schedule information is used to build a so-called time-dependent multimodal network.

1.5. Graph builder

Calculating routes in OPT is based on the navigation graph. OTP contains a component named the GraphBuilder to build a graph from different data sources. The Graph built is stored in a custom format that can then be accessed by OPT. This section briefly describes the GraphBuilder component, the graph model, the existing modules and the modules that have been added for i-Locate to add support for indoor routing.

The main graph builder process will scan a directory with files and configuration files and uses these to generate the graph. It will detect the file type from the extension and, depending on the extension, a different dedicated Graph Builder Module will be called to load the graph data and add it to the internal OTP Graph. Table 3 shows the file formats that are supported and handled by the different Graph Builder Modules.

Table 3: Graph builder modules in OTP for outdoor routing

Extensions	File format	Module
*.zip	GTFS	GtfsModule
*.pbf, *.osm, *.osm.xml	OSM	OpenStreetMapModule
*.tif	DEM	ElevationModule
-	-	TransitToStreetNetworkModule

When a specific file (extension) is found, the Graph Builder will make an instance of the matching module. Then it will run all modules to build the graph. After running these importer modules, which read the main road network data, some additional modules will be run, like the modules to connect the transit network to the road network (TransitToStreetNetworkModule).

1.5.1. Indoor graph builder

Graph Model

The navigation Graph in OTP is composed of Vertexes and Edges. The graph is directed, meaning that the Edges can only be navigated in one direction. Hence, for a two way street, two edges will have to be defined. The edges go from the fromVertex to the toVertex. The edges also contain the traversal methods. The traversal method determines how the edges are traversed and what the cost of the traversal will be. Together with the routing algorithm this will take care of things such as going uphill or first navigating bigger street before going to smaller street. But also the transport mode can be handled by the traversal algorithm of the Edges.

OTP already contains a lot of different Vertex and Edge types for different situations. The most important Vertex type is StreetVertex. But this does not add much to the standard Vertex. For the public transport part of the Graph, the TransitStopVertex contains more information about the stop. The most important Edge type is the StreetEdge. This contains all information about streets and how to traverse them.

Indoor Graph Model

We have added some classes to the OTP graph model to support indoor routing. We have chosen not to extend the StreetVertex and the StreetEdge but to introduce new base classes for indoor routing named IndoorVertex and IndoorEdge (extended from Vertex and Edge). In this way, it is always clear when we are doing something specific for indoor routing in the OTP code. The consequence of this decision is that at a lot of places in OTP code had to be duplicated because OTP is very much focused on outdoor routing and similar code was needed for indoor rooting. The design of OTP has a lot of room to make the routing general, but the current implementation obviously focuses on outdoor routing.

The main addition for the IndoorVertex is the level attribute, to indicate the level of a node. We added the following Vertex types:

- IndoorVertex – general indoor vertex and base class for the other indoor vertex types;
- IndoorDoorVertex – indoor vertex that represents a door;
- IndoorElevatorVertex – indoor vertex that represents a node that the elevator uses.

The IndoorEdge adds everything needed to let the indoor routing work. It doesn't have a level attribute because the vertices contain that information. The level of the start and end vertex of an edge can be different, like for a stairs or ramp. We added the following Edge types:

- IndoorEdge – general edge and base class of the other indoor edge types;
- IndoorElevatorEdge – an edge that represents the path on which the elevator moves;
- IndoorStairsEdge – an edge that represents a stair.

Indoor Graph Builder Modules

To support indoor routing, we added a few more indoor graph modules (Table 4). The IndoorGML is for reading graphs specified in IndoorGML. The AnchorNode Module is used to connect the indoor graphs together and to connect the indoor graphs to the outdoor network.

Table 4: Extended graph builder modules for indoor routing

Extensions	File format	Module
*.gml	IndoorGML	IndoorGmlModule
-	-	AnchorNodeModule

For each *.gml file in the source directory, an instance of IndoorGmlModule is created. And one AnchorNodeModule instance is created and added after all the indoor graph reader modules.

IndoorGmlModule

The IndoorGMLModule reads an IndoorGML file. This file should adhere to the specification as determined for the i-Locate project. This module does not check the validity of the file. It has been implemented in the

following way. It reads the XML file as a DOMDocument. Using XPath expressions it searches for the needed components to build the graph. Then it builds the graph using this information.

The module makes assumptions about how the geometry will be specified. This matches the way the portal generates the IndoorGML files.

Some special attributes have also been added for i-Locate to State elements, e.g. if a State (Vertex) is a door or an anchor node. In a similar way, the Transition elements can have a transitionType attribute, indicating if a transition is a stair, elevator or general indoor edge.

AnchorNodeModule

The AnchorNodeModule is used to connect sub-graphs together. It implements the following process.

1) Search for an anchor node, t
2) Determine the specific sub-graph this anchor node belongs to
3) Search for all other anchor nodes in this sub-graph
4) For each of the anchors in this sub-graph, search for the closest node outside this sub-graph. We call these nodes connecting nodes.
5) When a connecting node is found for each anchor node, add two FreeEdge's between the anchor and the connecting node (bi-directional connection).

The FreeEdge is a special edge that has no costs to traverse. It is only used to connect the different sub-graphs together.

This algorithm can connect different levels of a building together, but can also be used to connect the indoor graph to the outdoor graph. Thus, a unified graph is generated that can be used to navigate between all floors of a building and in and out of the building onto the outdoor graph.

Drawing the indoor graph

Fig. 1 systematically shows how a graph for a building with multiple floors can be drawn. The level of the Vertices are shown as the height. If two nodes are drawn very close together they are assumed to be in the same location. The figure also shows a proposed way of how to draw the graph.

Each level of the building can be drawn as a separate graph. Anchor nodes will always connect from a higher level to a lower level (this is a convention). So to connect level 1 to level 0 (L=1, L=0), an anchor node on level 1 will connect to the stairs that is part of level 0. Also the indoor graphs are the one that will connect to the outdoor graph. The stairs going from level 0 to level 1 are all part of the graph of level 0 (this is a convention).

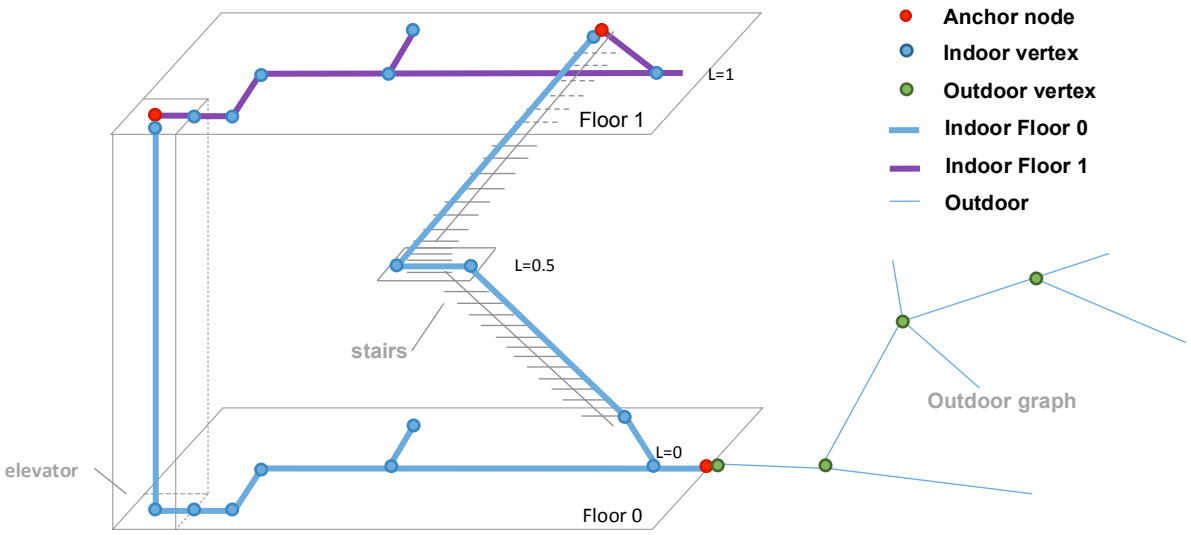

Fig. 1: Method of building an indoor graph

1.5.2. Outdoor graph builder

The outdoor graph is built using the street and public transport networks. The graph builder reads the road network data in the format of OSM and/or ESRI shape data. In combination with the GTFS data, the graph builder builds the integrated graph by creating edges and vertices for the specific types of road segments and virtual connection links. To build the outdoor navigation graph, it is necessary to provide the directory where the graph data are stored. OTP identifies the supported format of data automatically, e.g. OSM, shape data, GTFS.

The outdoor navigation graph is built based on the principle of a time-dependent network in the sense that departure and arrival times at nodes in the graph are bounded by the time schedules of transit services. In general, an edge represents a street segment or public-transport connection. For each street, two edges are created for the two directions, one in each direction. For each edge, a certain mode of travel is attached. It is possible that the transport modes on different directions of the same street are different, e.g. one-way street, bus line. For example, it is common in city centres that a two-way road (one direction for each) allows car in one direction and bus only in the other direction.

The road network is connected with the public transport network through transit stops or stations. Each stop/station is presented by at least one node. To connect the transit nodes with road networks, additional edges for boarding, alighting, dwelling and hopping are created. A hopping edge is used to connect two transit stops. For each stop or station, two vertices are created additionally to represent arrival vertex and departure vertex. The pre-board and pre-alight edges are created to connect these two types of vertices with the transit stop/station vertex. For each transit line, the vehicle vertices are created for arrival and departure respectively. Then, an alight edge and a board edge are created by connecting the vehicle vertices with the arrival and departure vertices. The dwell edge here is used to connect the two vehicle vertices (vehicle arrival and vehicle departure).

The Board, Alight, Dwell, and Hop edges represent transfer activities. These additional edges have weight functions depending on the time at which they are used. For example, a boarding edge has a waiting time depending on the next departure time of the public transport service under concern. In this way, the integrated graph is time-dependent, representing the information of all time tables involved.

9 Generating routes

Given a routing request, the routing system will produce the optimal route which takes into account preference parameters included in the request. The routing response is a trip plan which embeds the turn-by-turn directions for navigation guidance. In this section, we describe the route planning function and the design of turn-by-turn directions in the context of indoor and outdoor routing.

1.6. Indoor routing

1.6.1. Route planning

For indoor routing the same least-cost path finding algorithm is used as outdoor routing. The well-known A-STAR algorithm is used. Taking into account the specifics of indoor routing, the routing system considers the weight factors for special edges, i.e. stairs and elevators, and identifies whether an edge is traversable in case of a special transport mode, that is, using a wheelchair.

The default mode of travel assumed for indoor routing is walking. In that sense, the indoor routing model is a special case of routing with walking as transport mode. Other modes of indoor travel, like moving with wagons, are assumed to have the same speed as walking.

The distance and travel speed of an edge are used to calculate the general costs (travel time). In general, the length of indoor edges is the length of the shape. For walking and wheelchair a same speed of 3.2 km/hr is assumed. However, it is necessary to deal with special cases like stairs and elevators. Because stairs and elevators connect two floors in three dimensions, the length is a distance in three-dimensional space.

For stairs an additional weight parameter to take into account the extra effort involved compared to the elevator is used. Somewhat arbitrarily the costs of using a stair is assumed to be twice the costs of a general indoor edge for a same distance. Similarly, the costs of using an elevator are half of the costs of a general indoor edge of the same length. The length of the elevator between two adjacent floors is three meters as a default.

In the case of using a wheelchair, a route calculated should avoid the use of stairs or a ramp with a large slope. The property of whether an indoor edge can be traversed is considered in finding the appropriate route. In the i-Locate routing system the default value for whether an edge is wheelchair accessible is set dependent on the type of the indoor edge and slope. For general indoor edges or elevators, the default value is true, indicating that it is accessible. For edges of the stair type or that have a big slope, the default value is false. Below is the example code for stairs.

```
@Override
public boolean isWheelchairAccessible() {
  return false;
}
@Override
public boolean canTraverse(RoutingRequest options) {
```

```
 if (options.wheelchairAccessible) {
   if (!isWheelchairAccessible()) {
     return false;
   }
   if (getMaxSlope() > options.maxSlope) {
     return false;
   }
 }
 return canTraverse(options.modes);
}
```

1.6.2. Turn-by-turn directions

The indoor turn-by-turn directions provide guidance for the trips within buildings. As a part of the routing response, the indoor turn-by-turn directions are generated as auxiliary information of the route. The information for indoor directions is generated using the existing component in OTP for outdoor turn-by-turn direction with adjusted steps for indoor guidance.

For each edge of the route, the attributes of absolute direction and relative direction are calculated. The absolute direction defines the eight main directions as below.

```
public enum AbsoluteDirection {
  NORTH, NORTHEAST, EAST, SOUTHEAST, SOUTH, SOUTHWEST, WEST, NORTHWEST
}
```

The relative directions are defined based on the difference in absolute directions of the adjacent edges. Depending on the angle difference (relative direction), a message is selected. To give appropriate directions based on the angle difference, a list of relative direction types is defined, as below.

```
public enum RelativeDirection {
  DEPART, HARD_LEFT, LEFT, SLIGHTLY_LEFT, CONTINUE, SLIGHTLY_RIGHT, RIGHT, ARD_RIGHT,
  CIRCLE_CLOCKWISE, CIRCLE_COUNTERCLOCKWISE, ELEVATOR, GO_DOWN_WITH_THE_ELEVATOR,
  GO_UP_WITH_THE_ELEVATOR, UTURN_LEFT, UTURN_RIGHT, GO_UP_THE_STAIRS,
  GO_DOWN_THE_STAIRS, GO_THROUGH_THE_STAIRS, CONTINUE_THROUGH_THE_CORRIDOR,
}
```

For some information used in the outdoor navigation case, e.g. left, right, straight, we keep them the same as for indoor navigation. Special attention was given to the stairs and elevators. For example, in case of using a stair from one floor up to another floor, i.e. if the level of the start vertex is smaller than the level of

the end vertex of the edge, the step related to this will be GO_UP_THE_STAIRS. Depending on the direction of travel, additional steps are defined. The list of added walk steps for indoor navigation is shown in Table 5.

Table 5: Turn-by-turn direction for indoor navigation

	Steps
indoor navigation	ELEVATOR GO_DOWN_WITH_THE_ELEVATOR GO_UP_WITH_THE_ELEVATOR GO_UP_THE_STAIRS GO_DOWN_THE_STAIRS GO_THROUGH_THE_STAIRS CONTINUE_THROUGH_THE_CORRIDOR

1.7. Outdoor routing

1.7.1. Route planning

Outdoor routing has been well developed in OTP. Based on the time-dependent graph using road network data and public transport data, outdoor routing is also implemented using the A-star algorithm. The algorithm finds the optimal route with the least travel costs. Several preference parameters regarding the optimal route can be set. As a preference setting, either the departure time or arrival time of the trip can be set. The time information is used by the system to look up arrival and departure times given the schedules of public transport in the time dependent network. Thus, a different start time of the trips by transit will probably result in different optimal routes.

The list of possible travel modes considered in the OTP is shown in Table 6. Transit refers to a combination of bus and rail. All travel modes consider walking as a default connecting mode. As for the multimodal options, park & ride involves the use of car, park a car and the use of transit. The kiss & ride involves the use of car and transit without special parking stations for car parking. The bike & ride involves the use of bike, bike parking and use of transit.

Table 6: Travel modes in OTP

	Travel modes
Outdoor routing	walk, bike, drive, bus, rail, transit and multimodal modes of bike & transit, park & ride, bike & ride and kiss & ride

In case of bike only mode, OTP gives the option to choose a preference regarding a route. Three types of options are provided: the quickest, flattest and most bike friendly route. The default preference for a bike only trip is the weighted value of the three criteria, one-third for each. The factors related to time, safety and slope provide users further options to set preferences for a specific route. The sum of the three factors should always be equal to 1. These factors define additional weight to the existing weight function. For instance, if a person cares more about safety or slope when she/he is using bike, the safety or slope factor

can be given a larger weight than the others. If there is not a special setting of these factors, the routing algorithm will find the fastest route as a default without considering the weights.

1.7.2. Turn-by-turn directions

The outdoor turn-by-turn directions are also generated together with the trip plan. The directions are designed to provide guidance on cross-sections, e.g. continue, left. The direction information is generated for each edge of the route. The so-called walk step is used to define the text for turn-by-turn direction. In case of connected edges that have the same road name, only one walk step will be stored in the turn-by-turn information. A list of the walk steps used in outdoor turn-by-turn directions is shown in Table 7:

Table 7: Turn-by-turn direction for outdoor navigation

	Steps
Outdoor navigation	DEPART HARD_LEFT LEFT SLIGHTLY_LEFT CONTINUE SLIGHTLY_RIGHT RIGHT HARD_RIGHT CIRCLE_CLOCKWISE CIRCLE_COUNTERCLOCKWISE

1.8. Routing-system APIs

The routing service receives input parameters from clients as a routing request, finds the optimal route and sends the route plan together with corresponding turn-by-turn directions back to the client as a response. The request parameters are input to the routing API which is bundled as a Java Servlet on the web server. The routing algorithm requires time and location information from the client to find the specific optimal route. Furthermore, OTP uses an extensive list of input parameters to define more features of the optimal route. In reality, however, not all parameters need to be specified by users. With a short list of request parameters, the routing algorithm can find the specified route, while using default values for other parameters.

1.8.1. Routing requests

Because the i-Locate routing system integrates both outdoor and indoor routing, parameters of both parts need to be specified together in request. Table 8 shows the list of request parameters the routing system uses for integrated indoor-outdoor routing.

Table 8: Routing request (extended from OTP)

Request parameters	*Values*	*Meaning*
fromPlace	Longitude, latitude, level	Coordinates of the start location
toPlace	Longitude, latitude, level	Coordinates of the end location
date	Date	Start date of the journey
time	Time	Start time of the journey
departureFrom or arriveBy	Time	Departure or arrival time of the journey
mode	walk, bike, drive, bus, rail, transit and multimodal modes of bike & transit, park & ride, bike & ride and kiss & ride	Allowed transport modes
Optimize	Quick, safe, flat, greenways, triangle transfers	Parameters to optimize the routing algorithm
maxWalkDistance	Meters	Maximum distance of walking during the journey
wheelchair	Boolean	True if using wheelchair, false otherwise
showIntermediateStops	Boolean	True if show intermediate stops, false otherwise

The key parameters for a routing request are the start location, end location, date and time. The start and end locations are represented using longitude, latitude, and level. The level refers to the floor number. As explained before, the default value of level for outdoor locations is 0. Level information is needed because in an indoor environment, only longitude and latitude are not enough to identify a location uniquely.

The data and time information is used by the system to look up departure and arrival times in timetables of public transport. Users can choose to set either the departure or the arrival time. For indoor routing, the time information can be used to cope with various possible time restrictions, such as an appointed start time of a meeting or visit, opening and closing hours, etc.

In addition, the parameter Optimize specifies the criterion to be used for optimization. The list of possible optimization types a user can choose from is shown in Table 9.

Table 9: Parameters for optimized route

OptimizeType	Note
QUICK	The fastest trip
SAFE	The most safe trip
FLAT	The most flat trip.

GREENWAYS	The most user friendly trip.
TRIANGLE	The integration of the safe, flat and greeways.
TRANSFERS	Number of transfers

Optimize is an important parameter because it represents the preference and/or constraint one may have in reality. In case of outdoor routing for bike trips, it is possible for users to give the preference in terms of keywords, i.e. safe, flat and/or greenways. The safe keyword indicates the safety of street segments for bicycling. A standard street has a safety of 1. Safer streets have safeties between 0 and 1. The flat keyword refers to the level of slope. It reflects the slope of the ground, even if the road is a bridge above the ground, or a tunnel underneath it. The greenways is treated as a safer option than the default safe factor. For example, if a higher priority to greenways is given, a weight factor of 0.66 for bicycle safety will be taken instead of the default value, 1.0.

1.8.2. Routing responses

Routing response refers to the optimal route found by the routing service. It also includes a copy of the request parameters. The optimal route is not only a geometry object, but includes also details regarding time and attribute information. Therefore, the response parameters can be structured hierarchically using various objects. At the top of the response is a trip plan representing both a copy of the request parameters and a trip itinerary. The itinerary includes summary information of the route as a whole (transit time, waiting time, traveling distance, walking distance, arrival time, etc.) and of each route segment included (called Trip legs in OTP). In addition, the routing system also provides turn-by-turn directions in the responses. Such information is attached to the route segments (links), which is stored in the itinerary object under the header of Walk step. The list of response parameters is shown in Table 10.

Table 10: Routing response parameters (extended from OTP)

Response parameters	***Values***	***Meaning***
Trip plan	Date, fromPlace, toPlace, Itinerary	Object of the trip plan which includes sub-objects representing route details and a copy of request parameters
Itinerary	duration, startTime, endTime, walkTime, transitTime, waitingTime, walkDistance, walkLimitExceeded, elevationLost, elevationGained, transfers, fare, legs	Object of route representing detailed route information
Trip leg	startTime, endTime, departureDelay, arrivalDelay, realTime, isNonExactFrequency, headway, distance, fromPlace, toPlace, legGeometry, transitLeg, mode, route, agencyName, agencyUrl, agencyTimeZoneOffset, routeColor, routeType, routeId, routeTextColor, interlineWithPreviousLeg, tripShortName, tripBlockId, headsign, agencyId, tripId, serviceDate, intermediateStops, steps, notes, alerts, routeShortName, routeLongName, boardRule, alightRule, rentedBike, duration	Object of road link

Walk step	Distance, relativeDirection, streetName, absoluteDirection, exit, stayOn, area, bogusName, longitude, latitude, alerts, elevation, level	turn-by-turn information for navigation

Because the response parameters in the optimal route include the data of both indoor and outdoor parts, level is added to the responses of outdoor routing to represent the floor information of indoor trips.

Fig. 2 shows an example of the computed route using the integrated routing system. The route starts from an outdoor location and ends at an indoor location, with Walk only travel mode. The overview of the information of the route and turn-by-turn directions are shown on the right side.

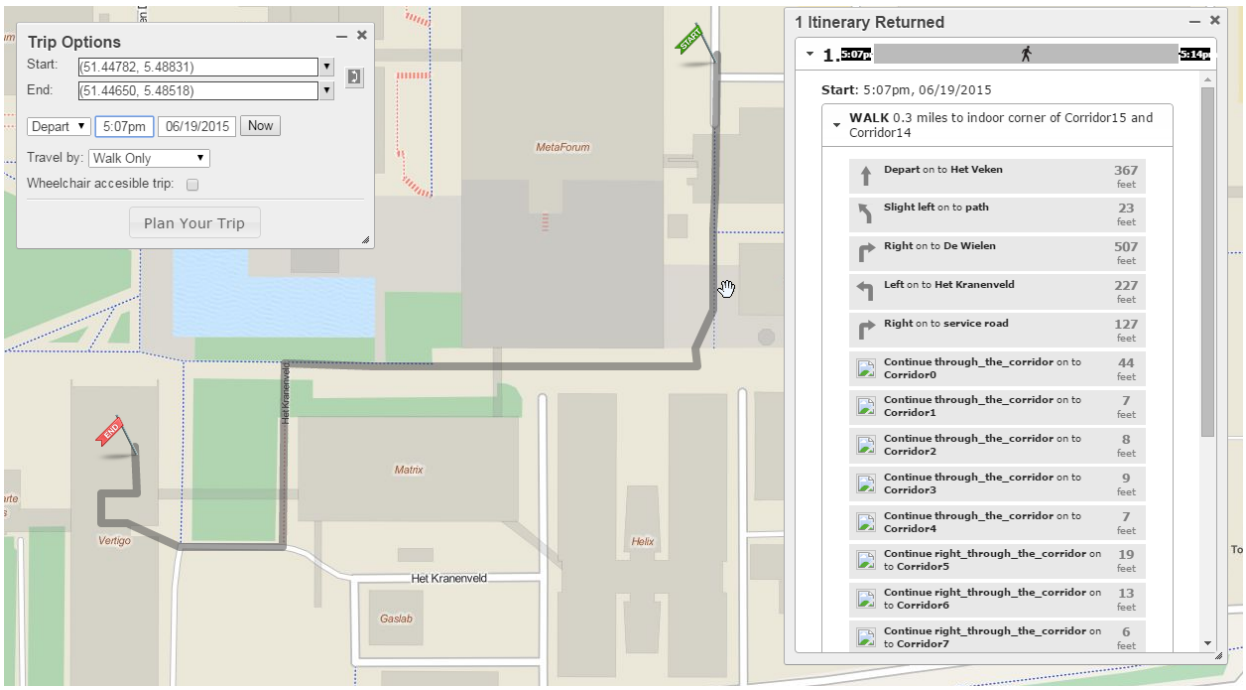

Fig. 2: Screenshot of the indoor and outdoor routing service

Below shows an example of the turn-by-turn direction when traveling between different floors within indoor environment. Assume the wheelchair is not used, the route directs to the use of stairs, involving seven steps.

Request parameters:

```
{
        time: "10:51am"
        showIntermediateStops: "false"
        arriveBy: "false"
        wheelchair: "false"
        maxWalkDistance: "804.672"
        fromPlace: "51.44624761721916,5.485031604766846,0"
        toPlace: "51.446254303950425,5.485101342201233,1"
        date: "06-04-2015"
```

```
        mode: "WALK"
}
```


Turn-by-turn directions:


```
steps:  [ 7]
- 0:  { distance: 2.755 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n4"
absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.48501499
lat: 51.44627426 level: 0 elevation:  [ 0] isDoor: false }
- 1:  { distance: 1.731 relativeDirection: "CONTINUE_THROUGH_THE_CORRIDOR" streetName: "n23"
absoluteDirection: "NORTH" stayOn: false area: false bogusName: false lon: 5.485054629
lat: 51.44627403 level: 0 elevation:  [ 0] isDoor: false }
- 2:  { distance: 4.616 relativeDirection: "GO_UP_THE_STAIRS" streetName: "Stairs23-n24"
absoluteDirection: "NORTH" stayOn: false area: false bogusName: false lon: 5.485061079
lat: 51.44628906 level: 0 elevation:  [ 0] isDoor: false }
- 3:  { distance: 1.274 relativeDirection: "CONTINUE_LEFT_THROUGH_THE_CORRIDOR" streetName: "n25"
absoluteDirection: "WEST" stayOn: false area: false bogusName: false lon: 5.485060331
lat: 51.44633055 level: 0.5 elevation:  [ 0] isDoor: false }
- 4:  { distance: 4.641 relativeDirection: "GO_UP_THE_STAIRS" streetName: "Stairs25-n26"
absoluteDirection: "SOUTH" stayOn: false area: false bogusName: false lon: 5.485042008
lat: 51.44633101 level: 0.5 elevation:  [ 0] isDoor: false }
- 5:  { distance: 1.911 relativeDirection: "CONTINUE_HARD_RIGHT_THROUGH_THE_CORRIDOR"
streetName: "n27" absoluteDirection: "SOUTHEAST" stayOn: false area: false bogusName: false
lon: 5.485042008 lat: 51.44628929 level: 1 elevation:  [ 0] isDoor: false }
- 6:  { distance: 2.471 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n3"
absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.485054629
lat: 51.44627403 level: 1 elevation:  [ 0] isDoor: false } - - }
```


With the same request parameters except that the wheelchair = true, the route directs to the use of elevator.

Request parameters:


```
{
        time: "10:51am"
        showIntermediateStops: "false"
        arriveBy: "false"
        wheelchair: "true"
        maxWalkDistance: "804.672"
        fromPlace: "51.44624761721916,5.485031604766846,0"
        toPlace: "51.446254303950425,5.485101342201233,1"
        date: "06-04-2015"
        mode: "WALK"
}
```


Turn-by-turn directions:


```
steps:  [ 15]
- 0:  { distance: 2.027 relativeDirection: "CONTINUE_LEFT_THROUGH_THE_CORRIDOR" streetName: "n5"
absoluteDirection: "WEST" stayOn: false area: false bogusName: false lon: 5.48501499
lat: 51.44627426 level: 0 elevation:  [ 0] isDoor: false }
- 1:  { distance: 5.885 relativeDirection: "CONTINUE_THROUGH_THE_CORRIDOR" streetName: "n6"
absoluteDirection: "NORTH" stayOn: false area: false bogusName: false lon: 5.484985822
lat: 51.44627403 level: 0 elevation:  [ 0] isDoor: false }
- 2:  { distance: 3.889 relativeDirection: "CONTINUE_THROUGH_THE_CORRIDOR" streetName: "n7"
absoluteDirection: "NORTH" stayOn: false area: false bogusName: false lon: 5.484985448
lat: 51.44632693 level: 0 elevation:  [ 0] isDoor: false }
```

- 3: { distance: 1.92 relativeDirection: "CONTINUE_THROUGH_THE_CORRIDOR" streetName: "n8" absoluteDirection: "NORTH" stayOn: false area: false bogusName: false lon: 5.4849847 lat: 51.44636189 level: 0 elevation: [0] isDoor: false }

- 4: { distance: 9.125 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n9" absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.484983578 lat: 51.44637914 level: 0 elevation: [0] isDoor: false }

- 5: { distance: 3.162 relativeDirection: "CONTINUE_LEFT_THROUGH_THE_CORRIDOR" streetName: "n27" absoluteDirection: "NORTHWEST" stayOn: false area: false bogusName: false lon: 5.485114835 lat: 51.44637867 level: 0 elevation: [0] isDoor: false }

- 6: { distance: 0 relativeDirection: "GO_UP_WITH_THE_ELEVATOR" streetName: "Elevator27-n28" absoluteDirection: "SOUTH" stayOn: false area: false bogusName: false lon: 5.485078184 lat: 51.4463955 level: 0 elevation: [0] isDoor: false }

- 7: { distance: 3.162 relativeDirection: "CONTINUE_HARD_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n28" absoluteDirection: "SOUTHEAST" stayOn: false area: false bogusName: false lon: 5.485078184 lat: 51.4463955 level: 1 elevation: [0] isDoor: false }

- 8: { distance: 9.125 relativeDirection: "CONTINUE_LEFT_THROUGH_THE_CORRIDOR" streetName: "n9" absoluteDirection: "WEST" stayOn: false area: false bogusName: false lon: 5.485114835 lat: 51.44637867 level: 1 elevation: [0] isDoor: false }

- 9: { distance: 1.92 relativeDirection: "CONTINUE_HARD_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n8" absoluteDirection: "SOUTH" stayOn: false area: false bogusName: false lon: 5.484983578 lat: 51.44637914 level: 1 elevation: [0] isDoor: false }

- 10: { distance: 3.889 relativeDirection: "CONTINUE_HARD_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n7" absoluteDirection: "SOUTH" stayOn: false area: false bogusName: false lon: 5.4849847 lat: 51.44636189 level: 1 elevation: [0] isDoor: false }

- 11: { distance: 5.885 relativeDirection: "CONTINUE_HARD_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n6" absoluteDirection: "SOUTH" stayOn: false area: false bogusName: false lon: 5.484985448 lat: 51.44632693 level: 1 elevation: [0] isDoor: false }

- 12: { distance: 2.027 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n5" absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.484985822 lat: 51.44627403 level: 1 elevation: [0] isDoor: false }

- 13: { distance: 2.755 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n4" absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.48501499 lat: 51.44627426 level: 1 elevation: [0] isDoor: false }

- 14: { distance: 2.471 relativeDirection: "CONTINUE_RIGHT_THROUGH_THE_CORRIDOR" streetName: "n3" absoluteDirection: "EAST" stayOn: false area: false bogusName: false lon: 5.485054629 lat: 51.44627403 level: 1 elevation: [0] isDoor: false }

10 Indoor geocoder

As additional support of indoor routing, an indoor geocoding service has been developed as part of this task (Task 3.3). By extending the existing outdoor geocoder, the indoor geocoding facility uses an extended indoor address to find the unique indoor location using the names provided. The identified coordination of the indoor location has the longitude, latitude and level information which can be used for indoor routing. The extended geocoder looks into the name properties provided by the IndoorGML file and extracts the level and coordination information.

There is the common issue of same names in different buildings. To avoid such ambiguityand keep the names unique, all names given in the IndoorGML file should include the name of the room and the name of the building.

1.9. Geocoder request

The geocoder is run as a service. Assume the local server is running on http://localhost, then the access point for the geocoding service is http://localhost/otp/routers/default/geocode. In this way, all OTP geocoders can be accessed, supporting outdoor and indoor geocoding. The request parameters can be given using the query string. Table 11 shows request parameters that can be used:

Table 11: Parameters of geocoder request

Request parameter	Value	Meaning
query	String (required)	String to geocode
autocomplete	Boolean (true/false) default false	Use query as prefix
stops	Boolean (true/false) default true	Search include public transport stops
clusters	Boolean (true/false) default false	Search for clusters by name
corners	Boolean (true/false) default true	Search for corners by at least one of the street names
rooms	Boolean (true/false) default true	Search for room names.

1.10. Geocoder response

Giving a geocoding request, the Geocoder will return the result encoded in JSON format. An example of a response is:

```
[{
    lat : 51.4463257690034,
    lng : 5.48493814293102,
    level : 1,
    description : " room VRT 1.3 "
  }, {
    lat : 51.4463611947343,
    lng : 5.4851144608899,
    level : 1,
    description : " room VRT 1.1 "
  }, {
    lat : 51.4463616608622,
    lng : 5.48517279726654,
```

```
    level : 1,
    description : " room VRT 1.2 "
  }, {
    lat : 51.4465854016838,
    lng : 5.48498582170039,
    level : 1,
    description : " room VRT 1.6 "
  }, {
    lat : 51.4462570150391,
    lng : 5.48493804944323,
    level : 1,
    description : " room VRT 1.4 "
  }, {
    lat : 51.4462375541525,
    lng : 5.48495207261069,
    level : 1,
    description : " room VRT 1.5 "
  }, {
    lat : 51.4463257690034,
    lng : 5.48493814293102,
    level : 0,
    description : " room VRT 0.3 "
  }, {
    lat : 51.4465854016838,
    lng : 5.48498582170039,
    level : 0,
    description : " room VRT 0.6 "
  }, {
    lat : 51.4463616608622,
    lng : 5.48517279726654,
    level : 0,
    description : " room VRT 0.1 "
  }, {
    lat : 51.4463611947343,
    lng : 5.4851144608899,
    level : 0,
    description : " room VRT 0.2 "
  }
]
```


For each 'record' in the response, the fields that will appear are shown in Table 12.

Table 12: Parameters of geocoder response

Request parameter	Value	Meaning
lat	Double	Latitude value of the location
lng	Double	Longitude value of the location
level	Double	If it is an indoor location the level field will be present giving the level of the location.
description	String	The name of the location, prefixed with the type of the location. Possible prefixes are: stop, corner, cluster or room.
id	String	For stops and clusters an id will be supplied.

11 Conclusions

In this deliverable document, we reported the development of the routing component as part of the i-Locate middleware. The routing service we developed here is based on extending OTP with specific added value to the indoor graph builder, routing, navigation and geocoder. The integrated routing service supports the use of IndoorGML data to build the indoor navigation graph. In combination with the outdoor road network and public transport data, an integrated navigation graph is built by connecting the indoor and outdoor graphs. The anchor node is defined as an entry point to connect indoor and outdoor graphs and the graphs of different floors of a building. Free edges are created based on the anchor node to build the connections between different graphs.

Based on the integrated navigation graph, the routing system gives the optimal route. Special attention has been given to the stairs and elevators, which have different weight factors when calculating the general costs on these special edges. Moreover, the issue of accessibility is also handled in case of the wheelchair in the sense that stairs, as a default, will not be accessible for wheelchair users.

To calculate the optimal route, the routing service requires different parameters which are set in a routing request, including start and end locations. The location information includes the level parameter which represents the floor number. In case of an indoor location, the indoor geocoder will extract the coordinates and level information using the given name (usually the name of a room).The level value for an outdoor location is zero as a default. As a result, the routing responses also include the level information for each edge of the route.

In addition, the turn-by-turn directions for indoor navigation have been specifically designed considering the distinct features of indoor navigation. Together with the existing turn-by-turn directions for outdoor routing, the directional information offers guidance to people to travel between indoor and outdoor environments cohesively.

The routing service developed in the i-Locate project is consistent with the open-source standard and is free to use by public. The data used to build the navigation graphs are also in the open data format. More importantly, the routing request and responses can be adapted to OpenLS standard. All these features make the routing system at the maximum extent benefit to the market.