A Semantic Data Model for Indoor Navigation

Liu Liu
GIS Technology, OTB Research Institute for the Built Environment, Delft University of Technology
Jaffalaan 9, 2628 BX Delft
The Netherlands
L.liu-1@tudelft.nl

Sisi Zlatanova
GIS Technology, OTB Research Institute for the Built Environment, Delft University of Technology
Jaffalaan 9, 2628 BX Delft
The Netherlands
S.Zlatanova@tudelft.nl

ABSTRACT
In this paper, we propose an indoor data model named Indoor Navigation Space Model (INSM). It is designed to support automatic derivation of the connectivity graph of a building. The INSM model provides an extended categorization of indoor spaces based on building semantics. It can be used to specify the nodes and edges of the connectivity graph. A UML class diagram of the INSM and a general approach of external data conversion are provided as well. An initial test of indoor path-finding is conducted and it demonstrates the feasibility of the INSM model to support indoor routing.

Categories and Subject Descriptors
H.2.1 [**Logical Design**]: Data models;

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms
Algorithms, Design

Keywords
Indoor Navigation, Building Semantics, Indoor Navigation Space Model, Path-finding.

1. INTRODUCTION
Indoor navigation is a task which consists of indoor localization, route planning and homing users on the designed routes. All the three stages need an appropriate representation of the indoor environments. Thus some sort of building model has to be constructed to facilitate indoor navigation. Generally, 3D topographic space is a fundamental aspect for indoor navigation. More specifically, the subspaces obtained from the whole building space with given semantics are significant to route planning.

The CityGML model and the Industry Foundation Classes (IFC) provide the topographic space of the indoor environment. They present geometric and topological relationships and certain semantics of the indoor environment. So they can potentially provide a part of or all of the necessary information for indoor navigation. Nevertheless, they are not specific models for indoor navigation. Some important information may not be explicitly and automatically obtained from them, such as the connectivity between staircases and other spaces. Yet this information is essential to indoor route planning, especially for graph-based approaches.

Furthermore, usually indoor obstacles can disturb the navigation process. However, state of the art researches and implementations of the indoor navigation (or the indoor path-finding) don't stress out the obstacle issue except for some simple emergency scenarios [9, 16]. For instance, in the CityGML building model a class named *IntBuildingInstallation* represents "an object inside a building with a specialized function or semantic meaning" [6]. Thus it can be regarded as a potential fixed obstacle, which means it can't be moved when pedestrians attempt to get through the space which it occupies. Another class *BuildingFurniture* can be considered as the potential moveable obstacles, such as chairs which can be budged. Nonetheless, there is no class to denote obstacles generated by hazards during emergencies.

Recently several navigation models have been investigated and reported in literature. The semantic model proposed in [14] mostly focuses on the *Granting* property of openings. It is not good enough for path derivation as it lacks the connectivity information between indoor spaces. The Combinatorial Data Model (CDM) is proposed in [10] to represent adjacency, connectivity and hierarchical relationships of building entities. It's a logical model and its representation is a pure graph without geometric properties. In order to apply network-based analyses (e.g. path-finding) on this model, the geometrical metric is introduced by means of Medial Axis Transformation (MAT) [1]. Finally, a geometric network is used to represent the building and support analyses. Another semantic model provided in [17] is also based on the geometric network of a building. But obstacles are not taken into account in the model. Recently a new semantic modelling method is reported in [18]. Yet it only aims at 2D geometries of buildings and the building semantics is quite simple. A "structured floor plan" is presented in [3]. It aims at supporting spatial design by getting semantic building spaces from the building geometry. It is also not a specific navigation model. For instance, the "beam" class is unnecessary to be used for path-finding.

The Indicative Route Method (IRM) is proposed in [8] on the basis of the "corridor map"[5]. The notion of "corridor" delineates the space that people can move freely among obstructions. The method inherently supports obstacle avoidance. Meanwhile, skeletons (the medial axes) or curves with certain clearance to the obstacles are smoothed. However, it is only a path-finding method and it doesn't involve the management of obstacles. If the free spaces are known to users, it implies the test scenario also has been known. So before applying this method to different kinds of scenarios, a model for scenarios is required.

ACM SIGSPATIAL ISA'12, Nov. 6, 2012. Redondo Beach, CA, USA,

The approach in [7] aims to construct the geometric-topological consistent indoor environment by making use of a grammar. Meanwhile, semantic information can be obtained as well. Though it's a promising way, much more rules have to be added in to reconstruct complex buildings.

A clear and very interesting model is proposed in [2]. Although the conceptual model grasps all key elements of buildings such as *IndoorObstacleSpace*, *TransitionSpace* and *IndoorSpace*, it's still a bit rough and it does not elaborate on the relations between the three key classes. Also, the means by which these elements group into the elements of higher level (e.g. *storey* and *IndoorEnvironment*) is vague.

The Multilayered Space-Event Model (MLSEM) [15] has been developed recently. It provides a fundamental framework for indoor navigation and allows links to be established between different layers such as building geometry, coverage of sensors *etc*. The research in this paper follows the general concepts of the MLSEM.

Based on the review given above, disadvantages of current navigation models can be summarized as follows:

- Limited link between indoor space subdivision strategies and path-finding approaches.
- Incomplete concerns about semantics of building spaces.
- Limited consideration of indoor navigation with obstacles. The routing algorithms mostly consider empty spaces. Yet in emergencies, obstacles are significant as they may change a path and impact the safety of pedestrian's movement.
- Little consideration of dynamically changing scenarios such as the emergency scenario.

In this paper, we propose a spatial-semantic coherent data model. It aims to address the above mentioned deficiencies. The model is specifically designed to support different environments/scenarios for specific indoor navigation tasks. It allows the connectivity to be automatically derived from the semantic hierarchy. Moreover, when applying classification/semantics of the model to indoor path-finding, we concentrate upon some "useful" building spaces and their characteristics.

The paper is developed as follows. In section 2, we will present our model and provide the definitions of indoor spaces. Section 3 will elaborate a general method to convert external data to the data in our model. Section 4 would address how our model is applied to indoor path-finding. Afterwards, section 5 will exemplify an initial test to demonstrate the feasibility of our approach. Ultimately, section 6 will conclude this paper with some future work.

2. Indoor Navigation Space Model

Generally, the purpose of space subdivision for indoor navigation is to automate the derivation of graphs of certain type [11, 14, 15]. But most of these graphs belong to the *Geometric Network*, i.e. metric information (e.g. coordinates) is assigned to each node of the graphs. Since the primary path-finding method is based on a graph representing building interior [4, 13], geometric routes can be calculated according to some rules and algorithms (e.g. the shortest way or the fastest way).

In this paper we consider the so called *Logical Graph* which doesn't involve metrics. Moreover, for indoor navigation we consider the *two-level routing strategy* which is presented in the previous publication [12]. The first level (i.e. the *rough* level) is based on an appropriate subdivision of a building. The *Logical Graph* (i.e. *connectivity graph*) is obtained on this level. On the basis of the *Logical Graph* the sequence of spaces to be followed is defined. At the second level the *Door-to-door* algorithm is applied. It utilizes metrics and considers obstacles to get a detailed path. Visibility graph (VG) is the foundation of the algorithm. A VG is constructed with the geometry of the current space and those of the obstacles contained in the space. Nodes of the VG involve the geometric vertices of obstacles and the concave vertices of the space; while edges represent the visibility between those nodes. After some shortest algorithm is applied to the VG, we can gain a shortest-distance and obstacle-avoided path. Yet in this paper, we further concentrate on the model of space subdivision. It's devised to bridge common-used standard datasets (IFC, CityGML and floor plans etc.) and indoor navigation.

2.1 Design Consideration

When we design a space model it is important to clarify the navigation spaces and their semantics. Also, the connectivity between these spaces should be readily identified. The principle is to effectively characterize indoor navigable spaces and keep the number of space types as less as possible. Thus we need to rethink about indoor spaces.

During the development of a novel model for buildings, we offer explicit indications of spaces. For a space these indications include its navigation usage, its connections to other spaces (i.e. multiplicity of connections), the possibilities to change floors and the free area within it. According to the usage of spaces, some spaces may be filtered and only the "useful" spaces will be collected for navigation. In the horizontal movement of users, certain transition spaces (e.g. openings) would be helpful as they connect different parts and control the accessibility of a building. In order to consider the vertical movement such like changing floors, we would like to extract some specific spaces to stairs and other vertical movement facilities (i.e. elevators).

For a space (a room or a corridor), it is important to know how many obstacles are in it. This can be an indication of the available free part for navigation. Furthermore, openings in a single space imply its connections with other spaces. The multiplicity of the connections may change the direction of movement.

According to various data of the indoor environment, for path computation we require a model which can identify a set of spaces and a set of relationships among the spaces from building semantics. The spaces are regarded as nodes and the relationships are considered edges. Then the graph derivation could be achieved automatically yet metrics (i.e. geometric information) are not introduced. The *space sequence* can be computed on this graph by applying non-metrics rules.

In the following section we will introduce the data model named Indoor Navigation Space Model (INSM). All the involved spaces (different building elements) are given with their formal definitions and relations. In addition, a UML class diagram of the model and a general external-data-conversion method will be discussed in detail.

2.2 Building Space Definitions

We use the notations of set theory to introduce our definitions. The introduced spaces and their relationships are also going to be

modelled with UML. We use capital letters to denote all sets and we take corresponding small letters to denote the set elements.

At first, we present three general indoor spaces. They are *Obstacle*, *Opening* and *Navigable Space cell*. Their definitions are given as follows.

1) *Obstacle (OBS)* is a space which cannot be entered by pedestrians.

2) *Opening (OPN)* is a transition space for the movement from one space to another space.

3) *Navigable Space cell (NSC)* is a space in which people can move freely without passing any *opn*.

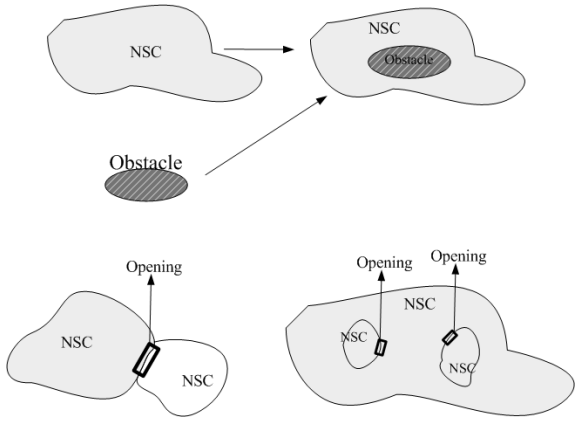

Figure 1. Illustration of general space notions

As shown in Figure 1, a *nsc* could contain one or more *obs*; and a *nsc* can contain one or more other *nsc* as well. A *opn* can only connect two *nsc*.

Some other basic building spaces are defined as follows:

4) *Vertical Unit (VU)* is a *nsc* in which people can move in vertical directions (up and down) along the same slope.

5) *Horizontal Unit (HU)* is a *nsc* in which people can move freely in horizontal directions.

6) *End* is a *hu* which is only connected to one another *nsc* at most.

Above five types (*OPN, OBS, NSC, VU, HU*) are the basic spaces in a building. Several other spaces will be defined with the help of them.

Here we introduce a useful auxiliary space type -- *Connector*. According to [14], the *Connector* represents corridors, elevators and stairs. A *Connector* has more than one entrance/exit [14]. In this paper we change its definition as an *End* also could have more than one entrance/exit. We define the *Connector* based on the number of its linked neighboring spaces. If a *nsc* is connected to at least two spaces $nsc_k, nsc_t \in NSC, k \neq t$, then the *nsc* is a *Connector (CON)*.

A *nsc*'s absolute bottom height is denoted by *botmh*, and its absolute top height is represented by *toph*. They are named the *nsc's heights (botmh, toph)* for short.

For a group of *nsc*, the maximum value of their *toph* is *MaxH* and the minimum value of their *botmh* is *MinH*. (*MinH, MaxH*) is a set of *Absolute Height Thresholds (AHTs)*.

$\forall nsc_k, nsc_t \in NSC, k \neq t,\ \exists r \in R,\ 0 < r < |toph_k - botmh_k| \wedge r < |toph_t - botmh_t|$, if $|botmh_k - botmh_t| < r \wedge |toph_k - toph_t| < r$, then *r* is the *Relative Height Threshold (RHT)*. In practice, the *RHT* is a given tolerance (e.g. 0.3 m).

Vertical Connector (VC) is a *hu* which connects at least two different *nsc*, meanwhile at least one of these *nsc* is a *vu*. That is, $VC := \{vc \in VC \mid \forall\ vc, \exists$ connected $nsc_k, nsc_t \in NSC, k \neq t : vc \in HU \wedge nsc_k \in VU \vee nsc_t \in VU\}$. We name it *VC* as it's a *con* and it connects at least one *vu*.

Horizontal Space Floor (HSF) is a group of *hu*. $HSF := \{hsf \in HSF \mid \forall\ hsf : hsf \in HU \wedge MinH < botmh < MaxH \wedge MinH < toph < MaxH\}$.

Contained Vertical Unit (CVU) is a group of *vu* contained in certain *hsf*. $CVU := \{cvu \in CVU \mid \forall\ cvu, \exists\ hsf \in HSF : cvu \in VU \wedge cvu$ is in $hsf\}$.

Dangling Horizontal Unit (DHU) is a group of *hu* which connects *HSF* to some *vc* but doesn't belong to *HSF*. $DHU := \{dhu \in DHU \mid \forall\ dhu, \exists$ connected $nsc_k, nsc_t \in NSC, k \neq t : dhu \in HU \wedge dhu \notin HSF \wedge nsc_k \in VC \wedge nsc_t \in HSF\}$.

Horizontal Space Vertical Connector (HSVC) $:= \{hsvc \in HSVC \mid \forall\ hsvc, \exists$ connected $nsc \in NSC : hsvc \in VC \wedge nsc \in HSF \bigcup DHU\}$.

Horizontal Space (HS, storey) $:= \{hs \in HS \mid \forall\ hs_t \in hs : hs_t \in HSF \bigcup CVU \bigcup DHU \bigcup HSVC\}$. *HS* is devised to represent the notion of "storey/floor" in reality. Originally it is the gathering of *hu* at certain height level. But in order to model the complex "storey" which contains intermediate-levels and some unequal-height spaces, we have adopted the above definition. Generally, a *hs* is the gathering of a group of *nsc*. The height level of a *hs* is controlled by a set of *AHTs* (*MinH, MaxH*).

Horizontal Connector (HC) is a *hu* which connects at least two other different *hu*, meanwhile all of them belong to a same *storey*. $HC := \{hc \in HC \mid \forall\ hc, \exists$ connected $hu_k, hu_t \in HU, k \neq t : hc \in HU \wedge hc, hu_k$ *and* hu_t *are in one hs*$\}$.

Vertical Connector in Vertical Space (VCVS) is a *hu* which only connects two different *vu*. Thus a *vcvs* is also a *vc*. $VCVS := \{vcvs \in VCVS \mid \forall\ vcvs, \exists$ only connected $nsc_k, nsc_t \in NSC, k \neq t : vcvs \in HU \wedge nsc_k \in VU \wedge nsc_t \in VU\}$.

Vertical Space (VS) generally is the aggregation of a group of *vu* and *vcvs*. $VS := \{vs \in VS \mid \forall vs_i \in vs : vs_i \in VU \bigcup VCVS\}$.

Building Space (BS) is the gathering of multiple *vs* and *hs* without *OBS*. $BS := \{bs \in BS \mid \forall bs_i \in bs : bs_i \in VS \bigcup HS\}$.

Building Part (BP) is the aggregation of multiple *bs*, *opn* and *obs*. $BP := \{bp \in BP \mid \forall bp_i \in bp : bp_i \in BS \bigcup OPN \bigcup OBS\}$.

Building (B) is the aggregation of multiple *bp*. $B := \{b \in B \mid \forall b : b \subseteq BP\}$.

From above definitions, several conclusions could be gained as follows. A *vu* connects at least two *vc*; a *vc* connects at least one *vu*; a *hc* connects at least two *hu*; and a *vcvs* connects only two *vu*.

Figure 2 illustrates some key building spaces of the INSM. They are *VC, VU, HC, HS* and *End*. In Figure 2, there is a *vu* contained

in a *hu*. Thus it's a *cvu*. And several *hu* on the ground constitute a *hsf*. On the left side of Figure 2, the top height of a connected *vc* exceeds those of the *hu*. According to the definition of *HS*, the *vc*, those *hu* and the *cvu* compose a *hs (storey)*. In this case, it demonstrates the capacity to describe and construct complex indoor environments (e.g. intermediate-levels in a storey and indoor spaces with distinct heights). Though a few models has already been proposed for indoor navigation [10, 14, 17, 18, 19], we believe the INSM is more capable of expressing most buildings without extra subdivision.

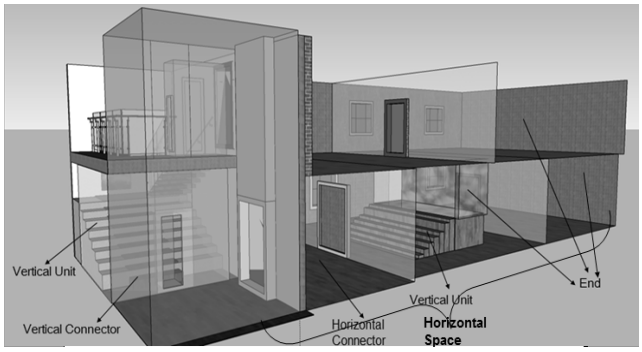

Figure 2. Building spaces as defined in INSM

The next section will present a UML class diagram of key building spaces which are defined in this section. Yet some supportive spaces such as *CON*, *HSF*, *CVU*, *DHU* and *HSVC* are not critical for navigation tasks. Thus they are not selected into the UML diagram.

2.3 UML Class Diagram

We name the data model as Indoor Navigation Space Model (INSM). The INSM model contains the basic spaces *OPN*, *OBS*, *NSC*, *VU*, *HU*, the specializations *End*, *VC*, *VCVS*, *HC*, and the aggregations *HS*, *VS*, *BS*, *BP*, *B*.

A fundamental class is *NSC*. It denotes all navigable units of a building. Thus it's the generalization of several other space classes. *HU* and *VU* are two basic subclasses of *NSC*. *HC* is the main horizontal passage between any two *hu* in a *hs (storey)*. *VC* connects *hu* / *vu* to *vu*, and vice versa. *VC* has a subclass *VCVS*. *VU* is the generalization of *Staircase*, *Escalator*, *Elevator*, *Ramp* and *VerticalTransferCells (*e.g. a spacious *nsc* spanning two *hs)*. *End* is a kind of *HU*. In general, *HC*, *VC* and *VU* are like vessels in a building. So path computation will stick to these main passages.

There are several other important classes in the diagram such like *OPN* and *OBS*. Except for *Door* and *Window*, a subclass of *OPN* named *TemporaryOpening* is designed to represent emergency exits. Also, the classes called *FacadeWindow* and *InteriorWindow*, the subclasses of *Window*, are distinguished according to their usages. For example, an instance of *FacadeWindow* at low-level *hs(storey)* can be used for evacuation when a scaling ladder is set to it.

The subclasses named *FixedObstacle* and *MoveableObstacle* are designed for the class *OBS*. Their instances (e.g. pillars and desks) can be found in normal situation. Yet another subclass called *DynamicObstacle* is devised to express some obstructions occurring along with building changes such as hazards or collapsing parts.

Furthermore, the class *VS* is aggregated by the classes *VU* and *VCVS*. Another aggregation class *HS* is contributed by *HU* and *VU*. *VS* and *HS* further aggregate to the class *BS*. Then *BS*, *OPN* and *OBS* are gathered to form the class *BP*. Finally, the class *B* is aggregated by *BP*.

Next we would elaborate the attributes of distinct classes in the UML diagram. The class *NSC* includes attributes as *TopHeight* and *BottomHeight* referring to a *nsc's heights*. It also has other pivotal attributes which are called *OpeningIDs*, *ContainedObstacleIDs*, *HazardPresence*, *ObstacleDensity and Crowdness*. The attribute *OpeningIDs* indicates which *opn* are related to a *nsc*. It will be used to support connectivity graph derivation of a building. The attributes *ObstacleDensity* and *Crowdness* denote the degree of obstruction and congestion in a *nsc*. They are used to compute the weights of edges on a graph for path-finding. Moreover, the attribute *ContainedObstacleIDs* is designed to imply which obstacles are contained in a *nsc*. The attribute *HazardPresence* indicates the current state of a *nsc*.

Both the classes *VU* and *VS* provide the attributes named *LinkTopStoreyID* and *LinkBottomStoreyID* to show their relations with different *hs*. In the class *HS*, the attributes *MaxHeight* and *MinHeight* represent the *AHTs* of related *hs* (*storey*). The information that a *hu* is contained in certain *hs* also should be stored. Thus the class *HU* is designed an attribute named *StoreyID*. As an *end* is connected to one *nsc* at most, so there is an attribute of the class *End* named *NeighborID* which indicates the connected *nsc*.

For the class *OPN*, the attribute *Existence* denotes whether a *opn* is existed; the attributes called *LeftSpaceID and RightSpaceID* provide the IDs of two connected *nsc*; *PassingDirections* reflects whether a *opn* is passable from a *nsc* to another one; The attribute called *IsExit* indicates whether a *opn* is a main entrance/exit of a building; Another attribute *CurrentState* reveals that a *opn* is opened or close. At last, in the class *OBS*, one of its attributes named *BelongedSpaceID* implies which *nsc* contains the *obs*.

So far we have presented the UML class diagram of building spaces according to the definitions in section 2.2. From conceptual viewpoint the UML diagram is in line with those definitions. But the UML diagram provides the better view on classes and relationships. Moreover, according to the UML diagram the INSM model can be directly prepared for implementation.

It's not enough to merely use the INSM for indoor navigation as available indoor data may be acquired from some other data model. Thus it's necessary to consider data conversion from other sources to the INSM. With the classes and their attributes of the INSM, the data conversion can be completed. In the next section, we will introduce the process.

3. Data Conversion to INSM

Sometimes there may be no obvious sign of certain space type in the INSM. For instance, a *vu* in Figure 2 is a staircase connected to a *vc*. There is a connection but no a genuine door between them. This gives us a hint that in reality we are not always lucky to have a complete dataset which totally conforms to the INSM model. So it requires an efficient and accurate way to convert the data from some other sources (e.g. CityGML or IFC models) and to import them to our INSM model.

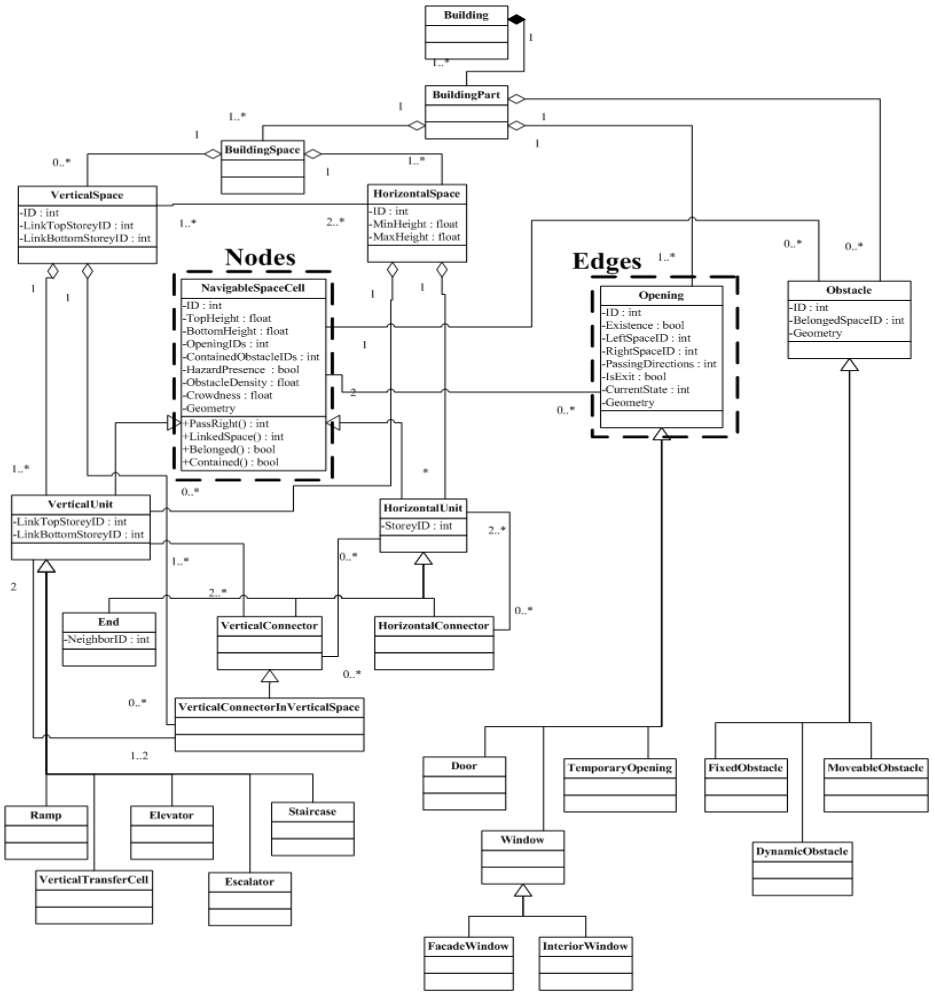

Figure 3. UML class diagram of INSM

In the following subsections, at first we introduce several specific operators of class *NSC*. Then we explain how these operators and semantics of the INSM are used to transform the data of external sources.

3.1 Operators

An operator named *PassRight()* of a *nsc* is defined to check the passing right of a *opn* of the *nsc*. Its input parameter is a *opn*. The return values are {*in, out, bi-direction, none*}. "*in/out*" means it's able to be in/out of a *nsc*, and "*bi-direction*" indicates one could get both in and out of a *nsc*. At last, "*none*' indicates a *nsc* can not be accessed by this *opn*.

As a *opn* connects two *nsc*, an operator of a *nsc* named *LinkedSpace()* is used to acquire the other *nsc* linked by the *opn*. Its input parameter is a *opn* and the return value is the connected *nsc*. That is, *opn* connects nsc_k, $nsc_t \in NSC$, $k \neq t$, $nsc_k.LinkedSpace(opn) = nsc_t$.

Another operator named *Belonged()* of a *nsc* is used to check whether the *nsc* conforms to a set of *AHTs* [*MinH, MaxH*]. That is, $\forall\ nsc \in NSC$, if $botmh \geqslant MinH \wedge toph \leqslant MaxH$, then *nsc.Belonged(AHTs) = true*; otherwise, *nsc.Belonged(AHTs) = false*.

The final operator of a *nsc* named *Contained()* is used to check whether the *nsc* is topologically contained in any another *nsc*. $\forall\ nsc_k$, $nsc_t \in NSC$, $k \neq t$, if nsc_t contains nsc_k, then $nsc_k.Contained(nsc_t) = true$; otherwise, $nsc_k.Contained(nsc_t) = false$.

3.2 Spaces Derivation of INSM

It's relatively easy to conduct the conversion of basic space types. For instance, the class "Room" in CityGML building model can be regarded as the class *NSC* in the INSM model. For simplicity, currently we suppose that several basic spaces (*OPN, OBS, NSC, VU, HU*) could be derived from external sources. Then we provide the method to obtain other spaces of the INSM. The detailed process is presented as follows.

End Detection: $\forall\ nsc \in NSC$, all of its related $opn_1,..opn_t \in OPN$, $t \geq 1$, if $nsc.LinkedSpace(opn_1) = \ldots = nsc.LinkedSpace(opn_t)$, then the *nsc* is an *end*;

Con Detection: $\forall\ nsc \in NSC$, $\exists$ related opn_i, $opn_j \in OPN$, $i \neq j$, if $nsc.LinkedSpace(opn_i) \neq nsc.LinkedSpace(opn_j)$, and at least $nsc.PassRight(opn_i) = in$, $nsc.PassRight(opn_j) = out$, then the *nsc* is a *con*. Specifically, it's the *con* to $nsc.LinkedSpace(opn_j)$. Here it's called *directed connector*. While if $nsc.PassRight(opn_i) = bi$-

direction $\wedge$ *nsc.PassRight(opn$_j$) = bi-direction*, then it's called *undirected connector*.

VC Detection: $\forall$ *hu* $\in$ *HU*, $\exists$ *opn$_i$*, *opn$_j$* $\in$ *OPN*, $i \neq j$, if *hu.LinkedSpace(opn$_i$)* $\in$ *VU* $\vee$ *hu.LinkedSpace(opn$_j$)* $\in$ *VU*, then the *hu* is a *vc*.

VCVS Detection: $\forall$ *hu* $\in$ *HU*, $\exists$ only *opn$_i$*, *opn$_j$* $\in$ *OPN*, $i \neq j$, if *hu.LinkedSpace(opn$_i$)* $\in$ *VU* $\wedge$ *hu.LinkedSpace(opn$_j$)* $\in$ *VU*, then the *hu* is a *vcvs*.

HSF Detection: $\forall$ *RHT*, *hsf* = {*hsf* $\in$ *HSF* | $\forall$ *hsf$_i$*, *hsf$_k$* $\in$ *hsf*, $k \neq i$: $|botmh_i - botmh_k| < RHT \wedge |toph_i - toph_k| < RHT$}. It's clear that *RHT* controls the number of *nsc* in a *hsf*.

CVU Detection: $\forall$ *hsf* $\in$ *HSF*, *cvu* = {*cvu* $\in$ *CVU* | $\forall$ *cvu$_t$* $\in$ *cvu*, $\exists$ *hsf$_j$* $\in$ *hsf*, *cvu$_t$.Contained(hsf$_j$)* = *true*}. It is obvious that an instance of *CVU* depends on a given *hsf*.

DHU Detection: $\forall$ *hsf* $\in$ *HSF*, *dhu* = {*dhu* $\in$ *DHU* | $\forall$ *dhu$_t$* $\in$ *dhu*, $\exists$ *opn$_k$*, *opn$_t$* $\in$ *OPN*, $k \neq t$: *dhu$_t$.LinkedSpace(opn$_k$)* $\in$ *VC* $\wedge$ *dhu$_t$.LinkedSpace(opn$_t$)* $\in$ *hsf*}. An instance of *DHU* also relies on a given *hsf*.

HSVC Detection: $\forall$ *hsf* $\in$ *HSF*, $\exists$ *dhu* $\in$ *DHU*, *hsvc* = {*hsvc* $\in$ *HSVC* | $\forall$ *hsvc$_t$* $\in$ *hsvc*, $\exists$ *opn* $\in$ *OPN* : *hsvc$_t$.LinkedSpace(opn)* $\in$ *hsf* $\cup$ *dhu*}. An instance of *HSVC* depends on a given *hsf* and its related *dhu*.

HS Detection: $\forall$ *RHT*, $\exists$ *hsf* $\in$ *HSF* $\wedge$ *cvu* $\in$ *CVU* $\wedge$ *dhu* $\in$ *DHU* $\wedge$ *hsvc* $\in$ *HSVC*, *hs* = {*hs* $\in$ *HS* | $\forall$ *hs$_t$* $\in$ *hs* : *hs$_t$* $\in$ *hsf* $\cup$ *cvu* $\cup$ *dhu* $\cup$ *hsvc*}.

HC Detection: $\forall$ *hs* $\in$ *HS*, $\exists$ *AHTs*, $\forall$ *hu* $\in$ *HU*, $\exists$ *opn$_i$*, *opn$_j$* $\in$ *OPN*, $i \neq j$, if *hu.LinkedSpace(opn$_i$)* $\in$ *HU* $\wedge$ *hu.LinkedSpace(opn$_j$)* $\in$ *HU* $\wedge$ *hu.Belonged(AHTs)* = *true*, then the *hu* is a *hc*.

VS Detection: *VS* = {*vcvs$_1$* $\cup$ *vcvs$_2$* $\cup$ … $\cup$ *vcvs$_m$*, *vcvs$_1$*,*vcvs$_2$*,..,*vcvs$_m$* $\in$ *VCVS*, $m \geq 0$} $\cup$ {*vu$_1$* $\cup$ *vu$_2$* $\cup$ … $\cup$ *vu$_n$*, *vu$_1$*,*vu$_2$*,..,*vu$_n$* $\in$ *VU*, $n \geq 1$}.

BS Detection: *BS* = {*vs$_1$* $\cup$ *vs$_2$* $\cup$ … $\cup$ *vs$_m$*, *vs$_1$*, *vs$_2$*,..,*vs$_m$* $\in$ *VS*, $m \geq 0$} $\cup$ {*hs$_1$* $\cup$ *hs$_2$* $\cup$ … $\cup$ *hs$_n$*, *hs$_1$*,*hs$_2$*,.., *hs$_n$* $\in$ *HS*, $n \geq 1$}.

BP Detection: *BP* = {*bs$_1$* $\cup$ *bs$_2$* $\cup$ … $\cup$ *bs$_m$*, *bs$_1$*, *bs$_2$*,.. *bs$_m$*, $\in$ *BS*, $m \geq 1$} $\cup$ {*opn$_1$* $\cup$ *opn$_2$* $\cup$ … $\cup$ *opn$_n$*, *opn$_1$*, *opn$_2$*,..*opn$_n$* $\in$ *OPN*, $n \geq 1$} $\cup$ {*obs$_1$* $\cup$ *obs$_2$* $\cup$ … $\cup$ *obs$_k$*, *obs$_1$*, *obs$_2$*,.. *obs$_k$* $\in$ *OBS*, $k \geq 0$}.

B Detection: *B* = {*bp$_1$* $\cup$ *bp$_2$* $\cup$ … $\cup$ *bp$_n$*, *bp$_1$*, *bp$_2$*,.. *bp$_n$* $\in$ *BP*, $n \geq 1$}.

So far the detection process is provided for the pivotal spaces defined in section 2.2. It lays a foundation to categorize building spaces for path-finding from certain building datasets.

All the semantics provided by the INSM is sufficient for indoor path-finding. The path-finding should be conducted in a prompt and reasonable way. In section 4, we will explain how the INSM model would be utilized to support indoor path-finding.

4. Path-finding with INSM

Based on the INSM and certain building data indoor path-finding can be realized. In this paper, we stick to *Logical-Graph*-based approach. With the INSM and the *two-level routing* strategy in [12], there is no extra subdivision on the *rough (first)* level (i.e. connectivity graph). Semantic and topological information of the building is utilized to build the connectivity graph at the *rough* level. After some rules and certain algorithm are applied to the connectivity graph, a space sequence indicating which *nsc* to be orderly passed will be obtained. We name the sequence as space sequence or *nsc* sequence. According to the semantics of distinct spaces, some patterns of the space sequence could be elicited. For instance, if the start and the destination are in a same *storey* (i.e. horizontal movement), a probable space sequence is *End -- HC -- End*. While a movement will occur between different stories, a probable sequence is *End – HC -- VC -- VU -- VC -- HC -- End*. It implies *VC* and *VU* would be concerned only in vertical movements.

The geometry of three kinds of space (*NSC, OPN and OBS*) will be concerned in the *detailed* (*second*) level. With some path-finding algorithm the final path can be computed. This path will be a geometric and obstacle-avoiding path.

The building spaces which are used for graph-construction (Nodes and Edges) are shown in Figure 3. Instances of *NSC* denote nodes and those of *OPN* represent edges. We can utilize only "valuable" *nsc* for the derivation of *Logical Graph*. For example, we can generate the graph without nodes of *End*. In this case, a *nsc*'s attribute *OpeningIDs* is checked to find the *opn* attached to it (see Figure 3). For each related *opn*, its attributes *LeftSpaceID* and *RightSpaceID* will be used to obtain the other related *nsc*. With the attribute *Passingdirections* of the *opn*, the edge's direction would be determined. For a *nsc* with more than one *opn* to another *nsc*, there is just one edge linking the two nodes of *nsc*. In this way, finally a directed connectivity graph will be built.

After the connectivity graph is acquired, some criteria are incorporated to provide weights on the edges of the graph, such as:

- Obstruction degree of *nsc*, including obstacle density and crowdness;
- The number of openings attached to *nsc*. We give priority to the *nsc* with more openings.

At this step metric information is not introduced to the weights on the edges of the graph. Afterwards, a certain type of algorithm (e.g. the shortest path) will be run on this graph. The computation result is a *rough* route represented by a *nsc* sequence.

When a *nsc* sequence is determined, a *detailed* route can be computed in each *nsc* by means of the *door-to-door* approach [12]. A graph is constructed on the basis of the geometric vertices of *obs* and *nsc*. The vertices of *obs* in the *nsc* and the concave vertices of the *nsc* compose the nodes of the graph. The visibilities between these nodes yield the edges of the graph. For each *nsc*, at first we need to check its attribute named "*ContainedObstacleIDs*" (see Figure 3) to get all obstacles in it. Finally, the shortest path algorithm is applied on the graph to get a geometric route in the *nsc*. Therefore, the geometric path is both the shortest and obstacle-avoiding. Until path-finding in every *nsc* on the sequence is finished, the entire geometric path is gained.

5. Initial Test

In this section we will demonstrate the usage of the INSM for indoor path-finding on a simple three-level building. A simple data model is implemented in a DBMS (Oracle 11g) according to the INSM. The spatial schema contains three tables named "*SpaceCell"*, "*Opening*" and "*Obstacle*". They respectively correspond to *NSC*, *OPN* and *OBS* of the INSM. The building data are populated in the DBMS as well. At present, we only consider 2D floor plans as the geometry of building objects.

5.1 Space Sequence

A connectivity graph can be automatically built based on the implementation of the data model and then path-finding will be conducted on it. In Figure 4, a connectivity graph is built with all *nsc*. The nodes with tag "2" represent *End*. Tag "0", "1" and "3" denote *HC*, *VC* and *VU* separately. The central part of Figure 4 is mainly about *vu* and *vc* and the three centrifugal clusters roughly indicate the three *hs*.

Figure 5 is the improved version of Figure 4. In Figure 5, there are merely two nodes with tag "2" (i.e. *end*). They represent start and destination spaces. Yet the other nodes are not *ends*. In this way the nodes of *End* are reduced and the routing is just conducted on main trunks of the connectivity graph.

The dark line in Figure 6 denotes a *nsc* sequence. The number labeled on each node is its ID. To each *nsc* on the sequence, their corresponding geometries would be retrieved to generate a geometric path. In the next section, an example will be given.

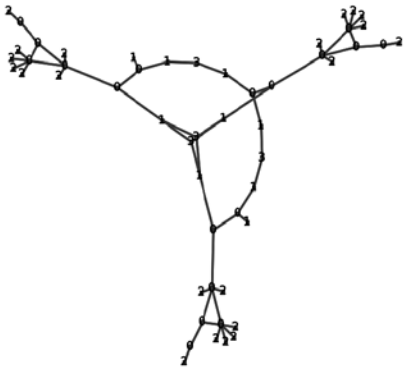

Figure 4. Logical Graph built with all Navigable Space cells

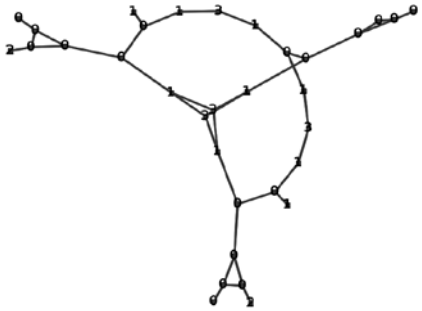

Figure 5. Logical Graph built without End nodes

Figure 6. Path from the start to the destination

5.2 Detailed Path

According to the *nsc* sequence in section 5.1, we compute geometric paths for all the *nsc* and we gather them to one detailed path. Figure 7 shows the path on the floor plans of the three-level building. It simulated a user started from the top floor, transferred by the elevator and arrived at the destination on the ground floor. Figure 8 demonstrates the geometric paths on the two floors in top view. The left image represents the top floor and the right one denotes the ground floor. The path on the ground floor shows obstacle-avoidance.

Test results in section 5 manifest the *Logical Graph* can be automatically derived from the INSM and the *two-level routing strategy* can be applied to getting detailed paths in a building.

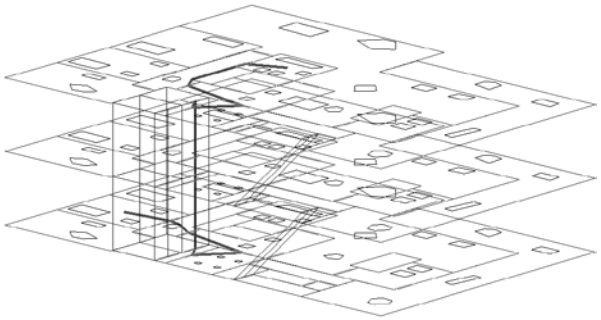

Figure 7. An entire detailed path on a pile of floor plans

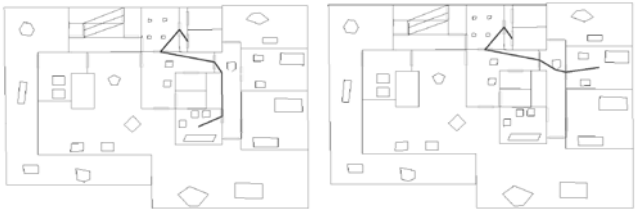

Figure 8. Detailed paths on distinct stories in top view

6. Conclusion

In this paper, we present a novel data model named Indoor Navigation Space Model to automate the derivation of the connectivity graph of buildings. We also provide a formalized method to transform the building data of other sources. Results from an initial test prove the usage of INSM for supporting indoor path-finding.

Our work is an extension of the concepts presented in MLSEM framework. The INSM provides further semantics specializations of building spaces in support of navigation. We believe the INSM lays a foundation for a generic way to represent the indoor environment. A building with complex interior could be decomposed for navigation according to the INSM while extra subdivision will not be introduced. In addition, the INSM model can support path-finding on both the *Logical Graph* and the *Geometric Network*.

INSM is designed to support the first level of path-finding according to the *two-level routing strategy*. It is quick and relatively simple to derive the connectivity graph (*Logical Graph*) from the INSM semantics. Currently only two criteria (obstruction degree and the number of openings) are used for the computation of space sequence. More criteria will be investigated in the future. The routing on the second level is based on the *Geometric Network* which is built in each *nsc* of the space sequence. Further research will address the options of employing the INSM model for the *two-level routing*. Routing results will be compared to those derived by other routing ways.

Though some specific navigation tasks can be conducted with specialized semantics of indoor spaces (such as searching *VC* and *VU* for vertical movements), the INSM lacks reasoning capabilities presently. So we may need to further extend the INSM to the task ontology for indoor navigation.

At present, the INSM is not validated yet by elaborated tests. In the next stage, we will validate the model with some real building data. Also, we need to consider the data conversion process according to the presented formalization. Special attention will be given on CityGML and IFC models.

7. REFERENCES

[1] Blum, H. 1967. A transformation for extracting new descriptors of shape. In *Proceedings of the Symposium on Models for the Perception of Speech and Visual Form*, Weiant Whaten- Dunn, Ed. MIT Press, Cambridge, Mass, 362-380.

[2] Brown G., Nagel C., Zlatanova S.,and Kolbe T. H. 2012. Modeling 3D topographic space against indoor navigation requirements. In *Proceedings of 3D GeoInfo 2012 Conference* (Quebec City, Canada, May 16-17, 2012).

[3] Choi J. W., Kwon D. Y., Hwang J. E., Lertlakkhanakula J. 2007. Real-time management of spatial information of design: A space-based floor plan representation of buildings. *Automation in Construction*. 16, 4 (July 2007), 449–459. DOI = http://dx.doi.org/10.1016/j.autcon.2006.08.003.

[4] Franz, G., Mallot, H. A., and Wiener, J. M. 2005. Graph-based models of space in architecture and cognitive science - a comparative analysis. In *Proceedings of 17th International Conference on Systems Research, Informatics and Cybernetics* (Windsor, Canada). 30-38.

[5] Geraerts, R., Overmars, M. 2007. The corridor map method: Real-time high-quality path planning. In *IEEE Int. Conf. on Robotics and Automation*. (2007). 1023–1028.

[6] Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K. 2012. *OpenGIS City Geography Markup Language (CityGML) Encoding Standard v2.0.0*. Open Geospatial Consortium Standard. Open Geospatial Consortium.

[7] Gröger, G. and Plümer, L. 2010. Derivation of 3D indoor models by grammars for route planning. *Photogrammetrie, Fernerkundung, Geoinformation*, 3:193-210. DOI: 10.1127/1432-8364/2010/0049

[8] Karamouzas, I., Geraerts, R., Overmars, M. 2009. Indicative routes for path planning and crowd simulation. *Foundations of Digital Games*. (2009), 113–120.

[9] Kwak, S., Nam, H. and Jun, C. 2010. An enhanced indoor pedestrian model supporting spatial DBMSs. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness* (San Jose, CA, USA, November 2, 2010). ISA'10. ACM, New York, NY, 25-32. DOI = http://doi.acm.org/10.1145/1865885.1865893

[10] Lee, J. 2004. A Spatial Access-Oriented Implementation of a 3-D GIS Topological Data Model for Urban Entities. *Geoinformatica*, 8, 3, 237-264.

[11] Li, X. Claramunt, C. and Ray, C. 2010. A grid graph-based model for the analysis of 2D indoor spaces. *Computers, Environment and Urban Systems*, 34, 6, 532-540.

[12] Liu, L., and Zlatanova, S. 2011. A "door-to-door" Path-finding Approach for Indoor Navigation. In *Proceedings of GeoInformation For Disaster Management Conference 2011* (Antalya, Turkey, May 3-8, 2011).

[13] Lyardet F., Szeto D.W., Aitenbichler E. 2008. Context-Aware Indoor Navigation. In *Proceedings of the European Conference on Ambient Intelligence* (Nuremberg, Germany, November 19-22, 2008). DOI = 10.1007/978-3-540-89617-3_19.

[14] Meijers, M., Zlatanova, S. and Preifer, N. 2005. 3D geoinformation indoors: structuring for evacuation, In *Proceedings of Next generation 3D city models* (Bonn, Germany, June 21-22, 2005).

[15] Nagel, C., Becker, T., Kaden, R., Li, K., Lee, J., and Kolbe, T. H. 2010. *Requirements and Space-Event Modeling for Indoor Navigation*. Open Geospatial Consortium Discussion Paper. Open Geospatial Consortium.

[16] Thill, J.C., Dao, T.H.D. and Zhou, Y. 2011. Traveling in the three-dimensional city: applications in route planning, accessibility assessment, location analysis and beyond. *Journal of Transport Geography*, 19, 3 (May 2011), 405-421.

[17] Tsetsos, V., Anagnostopoulos, C., Kikiras, P., Hasiotis, T.; Hadjiefthymiades, S. 2005. A human-centered semantic navigation system for indoor environments. In *Proceedings of International Conference on Pervasive Services* (July 11-14, 2005). ICPS '05. DOI = http://doi.acm.org/10.1109/PERSER.2005.15064

[18] Goetz, M. and Zipf, A. 2012. Mapping the Indoor World: Towards Crowdsourcing Geographic Information About Indoor Spaces. *GIM international*, 26, 3 (March. 2012), 30-34.

[19] Worboys, M. 2011. Modeling indoor space. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness* (Chicago, Illinois, USA, November 1, 2011). ISA'11. ACM, New York, NY, 1-6.