WIF4InL: Web-based integration framework for Indoor location

Long Niu Sachio Saiki Shinsuke Matsumoto Masahide Nakamura

WIF4InL: Web-based integration framework for Indoor location

Long Niu, Sachio Saiki, Shinsuke Matsumoto and Masahide Nakamura
Graduate School of System Informatics, Kobe University, Kobe, Japan

Received 6 September 2015
Revised 12 September 2015
6 October 2015
Accepted 1 February 2016

Abstract
Purpose – The purpose of this paper is to establish an application platform that addresses expensive development cost and effort of indoor location-aware application (InL-Apps) problems caused by tightly coupling between InL-App and indoor positioning systems (IPSs).
Design/methodology/approach – To achieve this purpose, in this paper, the authors proposes a Web-based integration framework called Web-based Integration Framework for Indoor Location (WIF4InL). With a common data model, WIF4InL integrates indoor location data obtained from heterogeneous IPS. It then provides application-neutral application programming interface (API) for various InL-Apps.
Findings – The authors integrate two different IPS (RedPin and BluePin) using WIF4InL and conduct a comparative study which is based on sufficiency of essential capabilities of location-dependent queries among three systems: RedPin, BluePin and WIF4InL. WIF4InL supports more capabilities for the location-dependent queries. Through the data and operation integration, WIF4InL even enriches the existing proprietary IPS.
Originality/value – As WIF4InL allows the loose coupling between IPS and InL-Apps, it significantly improves reusability of indoor location information and operation.

Keywords Indoor location query service, Indoor positioning framework, Integration framework, Location information, Location-aware service

Paper type Research paper

1. Introduction
With the rapid development of wireless and sensor technologies, *Indoor Positioning Systems* (IPSs) are attracting great attention in recent years. IPSs are systems that locate and track objects within indoor space (inside buildings, underground and so on) where Global Positioning System (GPS) does not work well. Compared to outdoor space, within indoor space, one can easily deploy extra infrastructure for the positioning system. There are many research and development of IPS in the past decade. Enabling technologies of IPS include sound or ultrasound (Harter *et al.*, 1999), image analysis (Wand *et al.*, 1992), RFID (Ni *et al.*, 2003), Wi-Fi (Tarrio *et al.*, 2011) and other radio-based approaches (Lorincz and Welsh, 2005). Although the goals of these IPSs are the same, they are different in many aspects, such as system topology, measuring principles, positioning algorithms and location information (data model).

By using IPS, various Indoor Location-aware Applications (we call indoor-location application or InL-Apps for short) can be implemented. An InL-App performs

This research was partially supported by the Japan Ministry of Education, Science, Sports and Culture [Grant-in-Aid for Scientific Research (B) (Number 26280115, Number 15H02701), Young Scientists (B) (Number 26730155) and Challenging Exploratory Research (15K12020)].

appropriate actions autonomously, according to the indoor location of users or dynamic/static objects. Typical use cases include searching goods or equipment in a hospital, locating of firemen in a building on fire, detecting the location of police dogs trained to find explosives in a building and finding tagged maintenance tools and equipment scattered all over a plant.

When we implement an InL-App with an IPS, it is necessary to determine, within the InL-App, how to represent and manage indoor-location data provided by IPS. Most conventional systems individually define and manage the indoor location information, considering the purpose of the InL-App and the characteristics of the IPS used. Such a proprietary representation and management method has an advantage of optimal performance. However, it causes "tight coupling" between InL-App and the underlying IPS, where indoor location data and operations cannot be reused among different InL-App. Thus, the proprietary method makes the implementation of InL-App complicated and increases development cost and effort.

As the first step to cope with problem, we have previously proposed a common data model for indoor location called Data Model for Indoor Locations (DM4InL) (Niu *et al.*, 2014). The DM4InL defines a common data schema for representing indoor location information of various objects (people, appliance, room, spot and so on) without depending on any specific IPS or InL-App.

Following the previous achievement of DM4InL, the main concern of this paper is how to construct a framework, where various InL-Apps can easily share and consume indoor location information gathered by various IPS. For this, we present the Web-based Integration Framework for Indoor Location (WIF4InL) in this paper. Using DM4InL, the proposed WIF4InL integrates indoor location data obtained from existing heterogeneous IPS and provides common operation as a service for various InL-Apps. Concerning this, we have to tackle two challenges. The first challenge is data integration, that is, how to convert indoor location data produced by heterogeneous IPS into DM4InL. The next challenge is operation integration, that is, how to implement comprehensive location-based queries retrieving data from DM4InL, to be shared by various InL-Apps.

To overcome those challenges, we design WIF4InL based on following three components:

(1) InL-Adapter;

(2) InL-Database; and

(3) InL-Query.

InL-Adapter adapts the proprietary indoor location data to common data model DM4InL. The InL-Adapter converts the uploaded location data into DM4InL. InL-Database is a large-scale shared database that manages the translated data. InL-Query provides application neutral application programming interface (API) for various InL-Apps to query the indoor location information. Operations for these components are published as cloud services, and thus, they are loosely coupled by service-oriented architecture.

To evaluate practical feasibility, we apply the proposed framework to integrate two different IPS. The first IPS is RedPin (Bolliger, 2008), which uses Wi-Fi fingerprints to locate mobile devices. The second IPS is BluePin, which uses Bluetooth beacons for detecting proximity of the devices. The proposed WIF4InL integrates the two different IPS, so that applications can transparently use indoor location information gathered by both systems. It is unnecessary for the applications to manage the difference of RedPin and BluePin. As WIF4InL allows the loose coupling between IPS and InL-Apps, it improves reusability and interoperability of indoor location information and operation. Thus, it is promising to reduce development cost and effort of InL-App, significantly.

2. Preliminaries

2.1 Indoor positioning systems

The IPSs generally refer to systems that estimate the position of subject or object within the indoor space. The primary progress in IPS has been made during the past ten years. Therefore, there is no *de facto* standard for the IPS yet, as compared to GPS. In general, IPS can divided into several categories:

- *Vision-based indoor localization*: Visual information can be collected and practiced for indoor navigation in many literature (Zufferey *et al.*, 2006; Mohamed *et al.*, 2011). However, the image-based localization will consume more computing resource (analyzing the image) and power.
- *Wireless-based indoor localization*: Unlike light, wireless wave can get through doors and walls and provide ubiquitous coverage of a building. The development with the existing wireless technologies (e.g. Wi-Fi, Bluetooth) is relatively easy, and the microwaves do not obstruct human activities in the building. Moreover, the power and computing resource consumptions are also significantly less than vision-based indoor localization. Most current work in indoor localization use this way.
- *Other methods*: There also many other ways for indoor localization. They include ultrasound (Harter *et al.*, 1999), acoustic background fingerprint (Tarzia *et al.*, 2011), accelerometer (Kothari *et al.*, 2012) and campus by adopting a dead-reckon method (Link *et al.*, 2011).

Among recent literatures, the wireless-based indoor localization methods take up most proportion of them. According to mathematical techniques used, they can be categorized into the following three groups:

(1) *Proximity*: This method assumes that if a user enters within the range of a known station, then the location of the user is approximated to the point of the station. We are currently developing an IPS, called BluePin, using Bluetooth Beacon technology. On detecting the proximity of a user, BluePin produces symbolic location data. The following data are symbolic location data. The following data L1 represent that user `P01` gets close to the entrance of the room S101. `L1: {personId: ‘P01’, locationId: 22, locationName: “S101 Entrance”, LastUpdate: 2015/07/27 11:23:45 JST}`

(2) *Triangulation*: This method uses geometric knowledge to obtain the user location. The location is determined by either the distance to the fixed known measurement points or the received signal angles.

(3) *Fingerprint*: The fingerprint means the characteristic of feature of signals. The method assumes that each position in the area has a unique fingerprint. Relying on prior knowledge associating a fingerprint with a position, the current location of a user is obtained. For example, Redpin (Bolliger, 2008) is

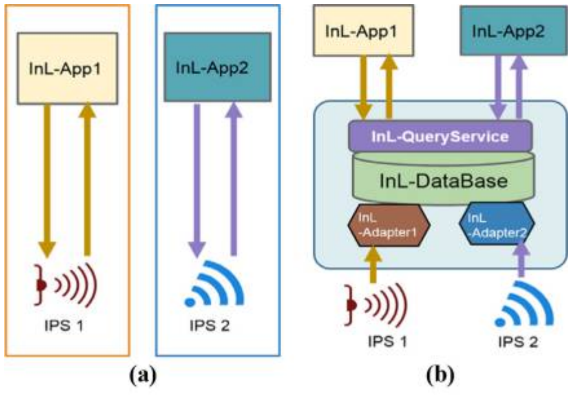
an open-source IPS which uses Wi-Fi fingerprint for zone-based positioning. The following data L2 are produced by RedPin, representing that a user is in location 45 of a room S103, pointed (345, 567) on a map System Building 1F: `L2: {locationId: 45, mapName: "System Building 1F", mapXcord: 346, mapYcord: 567, symbolicId: 'S103', macAddress: '08:60:6e:32:b6:0b'}`.

2.2 Indoor location application

In this paper, InL-App refers to any location-aware service or application that performs appropriate actions according to the indoor location information. To help understand, let us introduce the following examples:

- *SmartShop*: This service pushes coupons or loyalty program of a shop to a smartphone when a user approaches the shop. The user's location is estimated by BluePin, where a static beacon station is installed at the entrance of the shop. When a user `P01` gets closed to the station, the user's smartphone uploads the location data `{personId: 'P01', locationId: 7, locationName: 'shop1', lastUpdate: 2015/07/24 10:53:10 JST} to` a server. As SmartShop knows User 1 in the shop, it pushes shop coupons to the user's smartphone.
- *LocEyes*: This service visualizes locations of all staff working in an institute. We assume that every staff has a smartphone with RedPin and that the smartphone uploads the current indoor location every 10 s. An instance of the location is `{locationId: 45, mapName: "Building System 1F", mapXcord: 346, mapYcord: 567, symbolicId: "S103', macAddress: "08:60:6e:32:b6:0b'}`. According to the data, the server visualizes the latest location of every staff on the map Building System 1F.

It is easy to understand that there is no compatibility between SmartShop (with Bluepin) and LocEyes (with Redpin). Indeed, they are individually developed and operated, considering the service objectives and the underlying IPS. Figure 1(a) shows the



Notes: (a) Conventional InL-App; (b) proposed InL-App

Figure 1.
Two different architectures of InL-App

implementation architecture of these InL-Apps. We can see that each InL-App is tightly coupled with an IPS and that indoor location data and program are managed independently within each system. Therefore, one system cannot share or reuse the data and operation of another system. As a result, each InL-App has to be developed from scratch, which causes expensive cost and effort.

2.3 Previous work: Data Model for Indoor Locations

In our previous work (Niu *et al.*, 2014), we have proposed DM4InL. It defines a common data schema for representing indoor location information without depending on any specific IPS or InL-App. DM4InL consists of three models: location model, building model and object model. The location model represents spatial data in vector format and defines a global position for each building. The building model represents geographic elements, such as entrance, routes or room. The object model represents various mobile objects, such as human, robot or vehicle.

According to Yuan (1994), every spatial object must have theme, space and time attributes, to represent what, where and when, respectively. Hence, every object in the object model has an indoor-location point (in the location model), each of which is explained by spatial elements of a building (in the building model).

The original concept of DM4InL was published in a workshop paper. Changes were made on this paper, most significantly the addition of ObjectLocationLog, which stores the history of time-series indoor locations for every object. We will give some detailed descriptions of DM4InL in Section 3.3.

2.4 Long-term goal and scope of the paper

As mentioned in Section 2.2, the conventional architecture lacks compatibility and reusability, due to the tight coupling between InL-App and IPS. Therefore, our research goal is to establish an application platform as shown in Figure 1(b), which achieves loose coupling between InL-Apps and the underlying IPS. This paper focuses on implementing a framework that horizontally integrates the existing IPS and provides common operation as a service shared by various InL-Apps. To implement such a common framework, we have to tackle two challenges: *data integration* and *operation integration*. The data integration considers how to convert indoor location data produced by various IPS into the one conforming to a common data model. Following the previous work, we investigate how to convert the proprietary indoor location data into DM4InL. On the other hand, the operation integration deals with the implementation of comprehensive queries retrieving application-neutral location data from DM4InL.

3. Web-based Integration Framework for Indoor Location: integration framework for indoor location

3.1 Architecture

To overcome those challenges mentioned in Section 2.4, we propose WIF4InL (*Web-based Integration Framework for Indoor Location*). WIF4InL works as an abstract layer between InL-App and IPS. This layer first integrates indoor location data gathered by heterogeneous IPS and then provides application-neutral API for various InL-App, by which InL-App can access to different IPS transparently.

Figure 1(b) shows its architecture. The WIF4InL consists of three components: InL-Adapter, InL-Database and InL-Query. Features of each component are described below:

- *InL-Adapter (Indoor location adapter service)*: This is a Web service that adapts the proprietary indoor location data to the common data model DM4InL. When a client uploads proprietary indoor location data via Web-API, InL-Adapter converts the data into the one in DM4InL and inserts the converted data in a database (InL-Database, see below). As different IPS creates location data in different format, we need to implement a dedicated adapter for every IPS.
- *InL-Database (Indoor location database)*: This is a large-scale shared database that manages the indoor location data provided by InL-Adapter. Every record of indoor location data complies with DM4InL and is stored with time-stamp to keep the history.
- *InL-Query (Indoor location query service)*: This is a Web service for querying indoor location data stored in InL-Database. It provides application neutral API for various InL-Apps to query indoor location of any object. InL-Query provides two types of API: fundamental API and composite API.

The whole WIF4InL itself is deployed as cloud service, where the above components are loosely coupled by service-oriented architecture (SOA).

3.2 Approach overview

To manage the data integration and the operation integration, WIF4InL is designed specifically as follows:

- *Data integration*: Heterogeneous indoor location data are managed in a single schema of DM4InL. The conversion from proprietary data format into DM4InL is conducted by individual InL-Adapter. By doing this, it is unnecessary to modify the existing IPS. A client just uploads the location data via Web-API of designated InL-Adapter, where all the tasks for the data conversion and storing are delegated to WIF4InL. The detail of InL-Adapter will be described in Section 4.
- *Operation integration*: Heterogeneous operations for the existing IPS are consolidated by InL-Query, with which every InL-App can retrieve indoor location data in DM4InL. Each application does not need to know technical details of the underlying IPS. As will be shown in Section 5, InL-Query provides fundamental API and composite API.

3.3 Data schema of Data Model for Indoor Locations

Before going into the details, we briefly review data schema of DM4InL, as it is essential for WIF4InL. The DM4InL aims to prescribe a common data schema, independent of the implementation of IPS or the usage of InL-App. It represents the location of every indoor object with three kinds of models: location, building and object:

(1) *Location model*: It represents any location in a building by a relative position (three-dimensional offset) from the base coordinates of the building. Using the coordinate, we construct four geometric primitives: local point, local line, local polygon and local space. It also defines the global position in [longitude, latitude, altitude] and an angle formed by its *x*-axis and the north direction.

(2) *Building model*: It defines every building with theme attributes and global position. It also defines geographic elements in each building such as partitions, routes and spots. Each spot (route or partition) is located by a local point (a local line or a local space) in the location model. It also identifies every building with a reference point represented by a global position.

(3) *Object model*: It defines various objects in building, such as people, appliance, furniture and object location log stores location and time information of objects. Each object location log refers to a local point in the location model to represent its position.

Figure 2 shows an ER diagram of DM4InL, representing the relationships among the three models. The diagram follows the notation defined in the study by Watanabe (2003). A square represents an entity.

A relationship may be defined between a pair of entities, where:

- (+ − ϵ) represents a parent-child relationship;
- (+ − …) represents a reference relationship; and
- (+−°+) represents a sub-type relationship.

Every indoor location (i.e. LP, LLN or LSP) is associated with a single building, whereas every building involves more than one indoor location. A building (B) is located by a global position (GPos). A geographic element in a building (i.e. spot, route or partition) refers to a location entity (LP, LLN or LSP). An object location log refers to a local point and an object entity. The full description of DM4InL can be found in the study by Niu *et al.* (2014).

4. InL-Adapter for data integration

4.1 Overview

To achieve the data integration of heterogeneous IPS, the proposed WIF4InL implements InL-Adapter (*Indoor Location Adapter Service*). InL-Adapter is a Web service that adapts the proprietary indoor location data to DM4InL.

As mentioned in Section 3.2, clients of each type of IPS first upload the proprietary location data to InL-Adapter, and then, the InL-Adapter converts the data into the one with DM4InL.

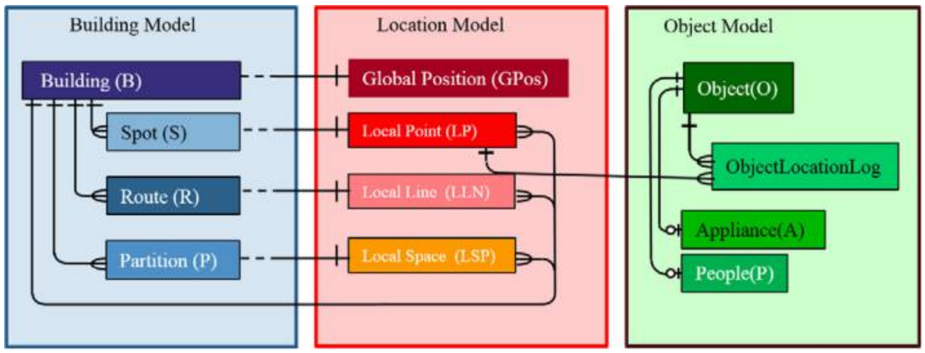


Figure 2. Data Model for Indoor Locations as a composition of three models

To implement this, we have to address two issues: *topology adaptation* and *data conversion*. The topology adaptation considers the structure of how to upload the measured data to InL-Adapter, which will be described in Section 4.2. The data conversion considers how the InL-Adapter converts the uploaded data into DM4InL format, which will be described in Section 4.3.

4.2 Topology adaptation

As a first step, we need to slightly modify the existing IPS, so that the measured indoor location data are uploaded to an InL-Adapter. For this modification, we have to consider the *system topology* of the IPS. However, the topology varies from one IPS to another. Therefore, we propose different adaptation patterns for different topology.

According to the study by Liu *et al.* (2007), there are four different system topologies for IPS: *remote-positioning*, *self-positioning*, *indirect remote-positioning* and *indirect self-positioning*.

Figure 3 shows the four topologies. In the figure, a triangle represents a static device or station deployed in the infrastructure. A circle represents a mobile device to be located. A rectangle represents a server. A hexagon represents an InL-Adapter, to which we newly adapt the existing IPS. The labels "M", "R" and "T" represent the roles of measuring unit, signal receiver and signal transmitter, respectively.

Figure 3(a) shows the remote-positioning topology, where the remote server locates the mobile device. The static stations receive the signal transmitted from the mobile device and forward the signal to the server. The server then computes the location of the mobile device. An IPS with presence sensors in the study by Kashio *et al.* (2015) belongs to this topology. In the remote positioning topology, all the location data are managed in the server. Therefore, we modify the server so as to upload the measured data to an InL-Adapter.

Figure 3(b) shows the self-positioning topology, where the mobile device itself measures the location. This mobile device receives signals from infrastructure and computes the current location from the signals. IMES (Manandhar *et al.*, 2010) and GPS

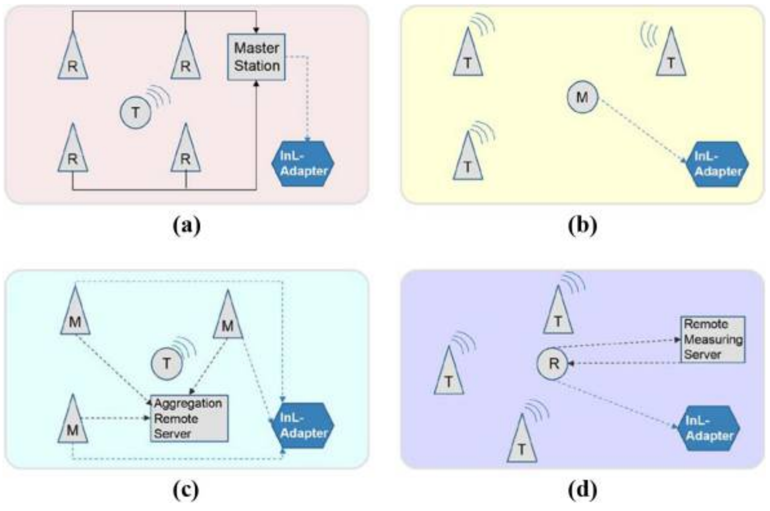


Figure 3.
Four different IPS topologies and adaptation patterns

belong to this topology. In the self-positioning topology, all the location data are managed by the mobile device. Therefore, we modify the mobile device so as to upload measured location data to an InL-Adapter.

Figure 3(c) shows the indirect remote-positioning topology, where the mobile device is located indirectly by the remote server. To cover wide area or multiple buildings, several stations with the measuring capability collaborate to send the location data to the aggregation server. In the indirect remote-positioning topology, the location data are managed by either the measuring stations or the aggregation server. Considering the fact that the aggregation server is often complex and is implemented with un-modifiable patent products, we choose to modify the stations to upload the data to an InL-Adapter.

Figure 3(d) shows the indirect self-positioning topology, where the mobile device indirectly obtains its location via the remote server. The mobile device first receives signals from the infrastructure and then forwards the signals to the remote server. The server computes the current location and returns the location data to the mobile device. With the development of IoT and cloud technologies, this type of IPSs are gaining popularity. RedPin and BluePin introduced in Section 2.1 belong to this topology. In the indirect self-positioning topology, the location data are held by either the mobile device or the measuring server. Considering the complexity and the workload of the measuring server, we choose to modify the mobile device to upload the returned location data to an InL-Adapter.

4.3 Data conversion

The second step is to consider how the InL-Adapter converts the uploaded location data into DM4InL. In general, every IPS defines its own location data format relying on the specific infrastructure and/or positioning algorithm. It is, therefore, impossible to enumerate data converters for all possible IPS in this paper. Instead, we present a *template* of how an InL-Adapter should convert the proprietary data into DM4InL. Then, the template is validated by practical examples with RedPin and BluePin.

Figure 4 shows the configuration template of InL-Adapter. As seen in the figure, the role of the InL-Adapter is to convert the measured location data in a proprietary format into the one in DM4InL. To achieve the conversion, two kinds of data depending on IPS are essential: *measured data* and *master data*. The measured data are real-time location data (i.e. raw data) measured by the given IPS. The master data are static data specifying various configuration information of IPS, such as users, devices, stations, buildings and indoor maps. As shown in Figure 4, every InL-Adapter contains *data converter*, which defines a specific mapping from the measured data into DM4InL based on the master data.

To help understand, we demonstrate the process of data conversion of BluePin and RedPin, using instances of location data L1 and L2 (see Section 2.1). According to DM4InL, we divide data items in L1 (or L2) into three elements: time, object and position. As for the time information, it is easy to introduce the common representation in UTC.

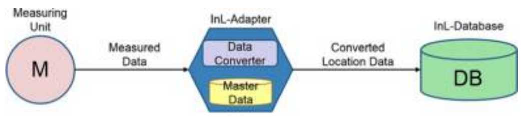


Figure 4. Configuration of InL-Adapter

For example, `“2015/07/27 11:23:45 JST”` in L1 can be converted into `‘2015-07-27T02:23:45’`. For L2, as RedPin does not define time attribute, we need to modify the RedPin client to add the times tamp to L2, like `‘2015-07-27T01:11:01’`.

As for the object information, we create a mapping from a proprietary ID of the located object into an object ID. For instance, the person ID `‘P01’` in L1 of BluePin is bound to object ID of DM4InL. Using the master data of BluePin, other data items of the object model can be filled. On the other hand, the macAddress in L2 of RedPin can be mapped to the object ID of DM4InL, as it is a unique string.

The conversion of the position information is rather complex. For L1, we need to convert the symbolic information `‘22’` and `“S101 Entrance”` into a spot in building model of DM4InL. Also, the spot should be represented by a local point. For this, we use the master data of BluePin to look up the detailed location information of 22. Suppose that the detailed information points position (3.50 m, 5.5 m, 1.5 m) of a building `‘B001’`. Then, we create a spot `“S101 Entrance”` in building `‘B001’`, and the spot’s coordinate is (3.50, 5.50, 1.50). Finally, we define a mapping from L1 to the spot.

On the other hand, as seen in L2, RedPin represents the position based on two-dimensional coordinate over a given map, that is, image of the floor plan. Therefore, multiplying the coordinates by the map scale derives the actual X and Y offsets. The Z offset can be derived from the altitude of the floor. Thus, the coordinates of a local point can be calculated. The spot information can be derived from the metadata of the floor map. For instance, suppose that `“System Building 1F”` represents a map of the first floor of building `‘B001’` with altitude of 1.5 m and that the map scale is 1/51.6. Then, L2 is converted into a spot bound to a local point (6.70, 10.44, 1.50). Based on the above conversion, the heterogeneous measured data L1 and L2 are converted into DM4InL format showed in Tables I, II and III.

Table I.
LocalPoint

Pcode	x-offset	y-offset	z-offset	Building-Seq
P001	3.50	5.50	1.50	B001-01
P002	6.70	10.44	1.50	B001-02

Table II.
Sopt

BuildingID	SpotID	SpotName	PointCode
B001	s00001	S101 Entrance	P031
B001	S00002	S103	P001

Table III.
ObjectLocationLog

ObjectID	P-Code	DateTime
P01	P001	2015-07-27T02:23:45
08:60:6e:32:b6:0b	P002	2015-07-27T01:11:01

4.4 Implementation

We have developed an InL-Adapter for RedPin and modified the client of RedPin. The modification of RedPin Android client comprised around 418 lines of code, and the InL-Adapter of RedPin comprised around 536 lines of code. Technologies used for the implementation are as follows:

- *Language*: Java 1.7.0;
- *Database*: MySQL 5.1;
- *Web server*: Apache Tomcat 7.0.57; and
- *Web service engine*: Apache Axis 2 1.6.2.

5. InL-Query for operation integration

5.1 Overview

To achieve the operation integration, the proposed WIF4InL implements InL-Query (*Indoor Location Query Service*). InL-Query is a Web service that provides application-neutral API for querying indoor location data stored in InL-Database. It is supposed to be deployed on cloud.

According to the data schema of DM4InL, we develop two types of API for InL-Query: *fundamental API* and *composite API*, as mentioned in Section 3.2 The fundamental API provides an interface for querying entities within a signal model at a time: location, building or object model. The details will be described in Section 5.2. The composite API allows advanced queries accessing multiple models simultaneously, which will be explained in Section 5.3.

5.2 Fundamental API

DM4InL represents location information of every indoor object with three kinds of models: location, building and object (see Section 3.3). Depending on the model to which a given query belongs, we define three groups of API: *location query API*, *building query API* and *object query API*.

5.2.1 Location query API. This API provides a set of methods (i.e. functions) querying any entity within the location model. As shown in Figure 2, the location model consists of four entities. Each entity has a set of methods that returns appropriate instances based on the given known attributes. The naming convention of the methods is `get[TargetEntity]By[given attribute]`. For instance, `getLPByPointCode(pointCode)` returns a local point designated by the given point codes.

The local query API also provides methods querying spatial relation among geometric primitives in the location model. The spatial relation can be used to investigate how a spatial object in a space is located in relation to another object. In Location Query API, we define two types of spatial relation:

(1) *Topological relation*: It represents how an object is topologically related to another object. The operators manipulating the topological relation include: `within, covers, coveredBy, intersects, touches, equals, disjoint, crosses and overlaps`. For instance, `getLPwitninLSP(LocalSpace)` returns all the local points within a given certain local space.

(2) *Distance relation*: It represents how far an object is from another object. The operators manipulating the distance relation include: `at, nearby, vicinity`

`and far`. For instance, `getLPnearbyLP(LocalPoint)` returns all local points nearby a given local point.

5.2.2 Building query API. This API provides methods for querying entities defined in the building model. To cover all possible queries, we derive the methods based on the structure shown in Figure 5. The figure shows that every method is constructed by varying *query entity* and *attribute items*. The query entity represents an entity to be returned by the method. Each entity has a designated set of attribute items, which explains the entity from the theme or spatial perspectives. Table IV summarizes the entities and attribute items contained in the building model (as well as in the object model). The methods are derived from all the possible combinations of the entities and the attribute items.

For instance, `getBuildingByGPID (GPID)` returns a building identified by a given global position ID. Also, `getRouteByName (buildingID, spotname)` searches routes in a given building by its name.

5.2.3 Object query API. This API provides methods querying entities in the object model. Similar to the Building Query API, the methods are constructed based on the structure in Figure 5. The attribute items are shown in the bottom of Table IV. We can see that the time attributes exists for the object entity, as an object usually takes a different location as the time passes. The methods are derived from all the possible combinations of the entities and the attribute items.

For instance, `getObjectsAtPoint (pointcode, dateTime)` returns a set of object that exists in a given local point on given data and time.

5.3 Composite API

The fundamental API allows only basic queries limited within a signal model. Hence, it is often too primitive to meet sophisticated requirements of InL-App, which requires

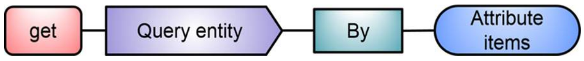


Figure 5.
Derivation process of building and object query API

Query entity	Attribute		Item
Building	Theme attribute		BuildingID, Name, Type
	Spatial attribute		GPID
Partition	Theme attribute		PartitionID, Name
	Spatial attribute		SpaceCode, BuildingID
Route	Theme attribute		RouteID, Name
	Spatial attribute		LineCode, BuildingID
Spot	Theme attribute		SpotID, Name
	Spatial attribute		PointCode, BuildingID
Object	Theme attribute	NULL	ObjectID, Type
		People	ObjectID, name, sex, . . .
		Appliance	ObjectID, Appliance Type, status, . . .
	Spatial attribute		PointCode
	Time attribute		Date Time

Table IV.
Detail entity and attribute item table

developers to integrate multiple API manually. For example, to implement a query "Who is in Room S101?", the developer needs to integrate the building query API and the object query API. This motivated us to develop the *composite API*, which allows high-level queries by internally combing some fundamental API.

Methods of the composite API have been derived based on typical use cases of location query in InL-App, so that they can reduce development cost and effort. Investigating the typical use cases, we have developed three type of composite API. The first type is *building-object* API, which returns geographic elements of a building based on known information of an object. For instance, `getPartitionContainObject(objectID)` returns a partition that contains a given object. The method is implemented by re-using multiple methods of the fundamental API, specifically:

```
Partition getPartitionContainObject(objectID)
  Object o = geObjectByID(objectID)
  LocalSpace ls = getLSPContainsLP(o.pointCode)
  Partition p = getPartitionByCode(ls.spacecode)
    Return p
```

The second type is *object-building* API, which searches objects based on known geographic elements of the building. For instance, `getPeopleWithinPartition(bName, pName)` returns people within a partition `pName` of a building `bName`. This method can be implemented as follows:

```
Person [ ] getPeopleWithinPartition(bName, pName)
  Partition p = getPartitionByName(bName, pName)
  LocalSpace ls = getLSPBySpaceCode(p.spaceCode)
  LocalPoint[ ] lps = getLPcontainedInLSP(ls)
  Person [ ] H = empty
  foreach ls in lps
    Person h = getPersonAtPointIntim(lp.pointCode, NOW)
    push(H, h)
  return H
```

The last type is *calculation* API, which measures a certain metric among object and geographic elements, using the spatial relations. For instance, `getDistanceBetweenSpotAndObject (buildingID, spotId, objectId)` returns a distance between a given spot and a given object. This method can be implemented as follows:

```
double getDistanceBetweenSpotAndObject(
  buildingId, spotId, objectId)
  Spot s = getSpotBySpotId(buildingId, spotId)
  Object o = getObjectByObjectId(objectId)
  double d = getDistanceBetweenLP(
    s.pointcode, o.pointcode)
  return d
```

5.4 Implementation

The implementation of the API is currently under way. Technologies used for the implementation are as follows:

- *Language*: Java 1.7.0;
- Database: MySQL 5.1;
- *Web server*: Apache Tomcat 7.0.57; and
- *Web service engine*: Apache Axis 2 1.6.2.

6. Evaluation

To evaluate the practical feasibility of WIF4InL, we conduct a comparative study among three IPS: RedPin, BluePin and WIF4InL (that integrates RedPin and BluePin).

6.1 Capabilities for location-dependent queries

The comparison is based on the sufficiency of essential capabilities of *location-dependent queries* (Ilarri *et al.*, 2010). The location dependent means that any change of the location of an object significantly affects the result of query for the object. For example, suppose that a User A wants to find friends within a range of 100 m from A while navigating a shopping center. The result of the query depends on A's current position, as well as on the location of the friends. According to the study by Ilarri *et al.* (2010), the following capabilities should be supported especially in the indoor location queries:

- *Position Queries* return the locations of mobile and static objects and are processed according to either a geometric or a symbolic model of space.
- *Navigation Queries* encompass all queries that directly help the users to find and reach some points of interest by providing them with navigational information while optimizing some criteria such as total traversed distance or travel time.
- *Range Queries* are used to find and retrieve information on objects of interest or places within a user-specified range or area.
- *k Nearest Neighbor(kNN) Queries* search for the k closest qualifying objects to a moving user with respect to his or her current location.

Moreover, Liu *et al.* (2007) suggested that the time is also an essential attribute for the location-dependent query.

- *Time queries* search a target object and a location by time or retrieve the time from an object and a location. Each query depends on a record that the object stayed in the location at the time.

6.2 Result of comparison

Table V compares the three IPS with respect to the above five capabilities. In the table, labels ○, △ and × represent that the capability is "satisfied", "partially satisfied" and

Table V.
Comparison of three IPS w.r.t capabilities of location-dependent queries

IPS	Position	Navigation	Range	kNN	Time
RedPin	○	×	△	△	×
BluePin	○	×	×	×	○
WIF4InL	○	△	○	○	○

"not satisfied", respectively. First, we can see that all the three IPS support the position queries. Although their representations of the position are different, they all have methods to ask the indoor location of a target object.

The navigation queries cannot be supported by RedPin or BluePin. In RedPin and BluePin, no topology information among multiple locations is maintained. Also, it cannot be derived from individual location data, as each location is represented as a pre-defined venue (not a position). However, once WIF4InL converts their location data into DM4InL, the topology can be defined using the coordinates of the locations. Using the topology, WIF4InL can support applications to implement navigation queries. Note, however, that the queries are limited to the spots or positions that are already registered in InL-Database.

WIF4InL binds indoor location with a three-dimensional coordinate in DM4InL. Using the topological and distance relations and the calculation API, one can easily implement the range queries. However, the range queries cannot be supported by BluePin, as BluePin specifies each location as a symbolic label, from which we cannot calculate the range. The location data of RedPin contain a two-dimensional coordinate on the map, from which we can calculate the distance between two locations. However, when the two locations are represented in separate maps (e.g. different floors), the distance cannot be calculated. In that sense, RedPin cannot fully satisfy the range queries. The same discussion applies to the kNN queries, as the essentials of the kNN queries are almost the same as the ones of the range queries.

As for time query, RedPin cannot support them as the data do not contain any time attribute, BluePin contains a time-stamp in the location data, while WIF4InL manages time-series data of ObjectLocationLog in DM4InL. Therefore, these two IPS can support the time queries.

Based on the above discussion, we can see that WIF4InL supports more capabilities for the location-dependent queries. Through the data and operation integration, WIF4InL even enriches the existing proprietary IPS. Thus, it is expected that the application developers can develop InL-App more efficiently and intuitively, using WIF4InL.

7. Related work

Our work is situated closely to the intersecting fields of indoor location framework, indoor positioning platforms and data modeling techniques for indoor spaces. A number of indoor positioning framework or platforms have been proposed so far.

Brachmann (2011) proposed a multi-platform software framework. It aims to manage sensor data from different smartphone platforms for better understanding of RSSI (Wi-Fi)-based and other sensor-based IPS. The key idea of this framework lies in the normalization techniques for individual sensor data, such as magnetometer, accelerometer and gyroscope. Thus, the framework is limited for wireless IPS, which belongs to indirect self-positioning system (Figure 3 and 4). It does not consider other IPS topology. In this sense, the application scope is narrower than WIF4InL.

Gubi *et al.* (2010) presented a platform that can dynamically provide efficient location technologies. As a user moves around a building, the platform suggests a best-available indoor positioning method based on the current position of the user. The platform manages building data in the form of symbolic map and markup of the associated RF infrastructure Wi-Fi and Bluetooth. However, the platform assumes that applications manage their own maps individually. So it does not provide application-neutral API that can re-use the indoor location data over different applications.

INSITEO Inc. (2014) presented an IPS technology that relies on optimal hybridization algorithms of multiple information sources. The data source includes power measurement of Wi-Fi, Bluetooth Low Energy signals and smartphone sensors (accelerometer, compass, barometer and so on). However, this approach is similar to Gubi's, which focuses on the combination of location technologies at the IPS level. Neither of them aims the loose coupling between IPS and InL-App.

8. Conclusion

In this paper, we have proposed a cloud-based integration framework, called WIF4InL, to achieve data and operation integration for heterogeneous IPS. To achieve the data integration, WIF4InL implements InL-Adapter which provides different adaptation patterns for different system topology of IPS. We also have implemented InL-Query, which provides fundamental API and composite API based on the data schema of DM4InL. WIF4InL contributes to loose coupling of IPS and InL-App, which will significantly improve the efficiency and re-usability in the InL-App development. We have actually applied the proposed framework to integrate two existing IPS, RedPin and BluePin. In addition, we have evaluated the WIF4InL by investigating the sufficiency of the five capabilities of location-dependent queries.

As for the future work, we plan to conduct further evaluation of WIF4InL, with respect to performance and security for practical use cases. We are also interested in how to address more pragmatic issues. They include uncertainty of location caused by unreliable devices, as well as feature interactions when integrating data and operations.

References

Bolliger, P. (2008), *Redpin – Adaptive, Zero-configuration Indoor Localization Through User Collaboration*, ACM, New York, NY.

Brachmann, F. (2011), *A Multi-Platform Software Framework for the Analysis of Multiple Sensor Techniques in Hybrid Positioning Systems*, IEEE, Busan, pp. 436-441.

Gubi, K., Wasinger, R., Fry, M., Kay, J., Kuflik, T. and Kummerfeld, B. (2010), *Towards a Generic Platform for Indoor Localisation Using Existing Infrastructure and Symbolic Maps*, Springer-Verlag, Berlin.

Harter, A., Hopper, A., Steggles, P., Ward, A. and, Websteret, P. (1999), *The Anatomy of a Context-aware Application*, ACM, Seattle, WA, pp. 59-68.

Ilarri, S., Mena, E. and Illarramendi, A. (2010), "Location-dependent query processing: where we are and where we are heading", *ACM Computing Surveys*, Vol. 42 No. 3, pp. 12:1-12:73.

INSITEO Inc. (2014), *Platform of Insiteo*, available at: www.insiteo.com/joomla/index.php/en/plateform (accessed 7 September 2015).

Kashio, Y., Matsumoto, S., Saiki, S. and Nakamura, M. (2015), *Design and Implementation of Service Framework for Presence Sensing*, IEEE, Moscom.

Kothari, N., Kannan, B., Glasgwow, E.D. and Dias, M.B. (2012), "Robust indoor localization on a commercial smart phone", *Procedia Computer Science*, Vol. 10 No. 2012, pp. 1114-1120.

Link, J.A.B., Smith, P., Viol, N. and Wehrle, K. (2011), *FootPath: Accurate Map-based Indoor Navigation using Smartphones*, IEEE, Portugal.

Liu, H., Darabi, H., Banerjee, P. and Liu, J. (2007), "Survey of wireless indoor positioning techniques and systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 37 No. 6, pp. 1067-1080.

Lorincz, K. and Welsh, M. (2005), "MoteTrack: a robust, decentralized approach to RF-Based location tracking", *Personal and Ubiquitous Computing*, Vol. 11 No. 6, pp. 489-503.

Manandhar, D., Kawaguchi, S. and Torimoto, H. (2010), *Results of IMES (Indoor Messaging System) Implementation for Seamless Indoor Navigation and Social Infrastructure Platform*, Institute of Navigation (ION), Portland.

Mohamed, M.K., Patra, S. and Lanzo, A. (2011), *Designing simple Indoor Navigation System for UAVs*, IEEE, Corfu.

Ni, M.L., Liu, Y., Lau, C.Y. and Patil, P.A. (2003), *LANDMARC: Indoor Location Sensing Using Active RFID*, IEEE, Fort Worth, TX.

Niu, L., Matsumoto, S., Saiki, S. and Nakamura, M. (2014), *Considering Common Data Model for Indoor Location-aware Services*, ACM, Shanghai.

Tarrio, P., Cesana, M., Tagliasacchi, M. and Redondi, A. (2011), *An Energy-Efficient Strategy for Combined RSS-PDR Indoor Localization*, IEEE, Seattle.

Tarzia, S.P., Dinda, P.A., Robert, D.P. and Memik, G. (2011), *Indoor Localization Without Infrastructure Using the Acoustic Background Spectrum*, ACM, Bethesda, MD.

Wand, R., Hopper, A., Falcao, V. and Gibbons, J. (1992), "The active badge location system", *ACM Transactions on Information Systems*, Vol. 10 No. 1, pp. 91-102.

Watanabe, K. (2003), *Introduction to Data Modeling for Database Designing*, 1st edn., Nippon Jitsugyo, Tokyo.

Yuan, M. (1994), *Wildfire Conceptual Modeling for Building GIS Space-Time Models*, GIS/LIS, Phoenix.

Zufferey, J.C., Klaptocz, A., Beyeler, A., Nicoud, J. and Floreano, D. (2006), *A 10-gram Microflyer for Vision-based Indoor Navigation*, IEEE, Beijing.

Corresponding author
Long Niu can be contacted at: longniu@ws.cs.kobe-u.ac.jp