To appear in the *Journal of Location Based Services*
Vol. 00, No. 00, Month 20XX, 1–18

ILONA: Indoor Localization and Navigation System

Zsolt Tóth *,

Institute of Information Science, University of Miskolc, Miskolc, Hungary ;

(*Received 00 Month 20XX; final version received 00 Month 20XX; accepted 00 Month 20XX*)

Indoor positioning is a hot topic these days and there is a growing need for it in public buildings such as airports, hospitals, universities or shopping malls. Indoor positioning systems should be accurate, easily available for the users, with low installation and maintenance cost, which makes development challenging. Existing systems are based on various technologies such as ultrasonic, RFID, WiFi or light encoding. Moreover, these systems are tailored to a given environment and usually rely on a single technology. This paper presents the ILONA System, a flexible hybrid indoor positioning and navigation framework. The ILONA System was not designed to be a solution for a single indoor positioning task but to be a standard core component of various systems. The ILONA System provides easily available positioning and navigation services for the end users. The system can manage data from the most commonly available sensors of modern smart phones. Thus, the ILONA System can perform positioning based on various technologies. ILONA System can be established at low cost because it only requires a connection between the server and the clients and WLAN is usually available. Hence, the presented ILONA System provides a widely available, hybrid indoor positioning framework at low cost to the developers of other indoor positioning solutions.

Keywords: Software Architecture, Indoor Positioning Systems, Indoor Navigation, Hybrid Positioning Methods

1. Introduction

Indoor positioning and navigation is a hot research area these days. The goal of these systems is to provide location information to users in a closed environment. Three tasks common in many real life scenarios are positioning, navigation and tracking. A positioning function allows users to query their position easily so they will not get lost in unfamiliar places. Orientation in a complex unknown building can be difficult, even if there are signs. A navigation function can be used to generate route between two known locations. The source location is usually determined by the positioning system; however, it is not necessary. Indoor navigation systems can help a person to find the shops in a mall, the gates in an airport, collect products in a supermarket or find a lecture hall at a university. Tracking could be used to query the current location of people or indispensable objects. The objects tracked use the positioning service to determine their location, which can be stored in either client or server side. This function would allow the redirection of the calls in a factory or it would ease finding a teacher in a university. Tracking of

*Corresponding author. Email: tothzs@iit.uni-miskolc.hu

essential objects, such as expensive tools in a factory, would increase productivity and increase the security of objects.

The development of indoor positioning systems dates back to the early 1990s, when the Active Badges system was developed at AT&T by Want and Hopper (1992). Since then, various solutions have been implemented that are based on different technologies such as Bluetooth, infrared, ultrasound and radio frequency. The existing indoor positioning systems usually use only one of these technologies, so their performance and applicability are limited by the chosen technology. Developers have to make a trade-off between accuracy and cost. Accurate solutions that can determine the position within a meter require the installation of specific hardware components, so they are costly. On the other hand, systems that use the commonly available infrastructure have lower installation costs but they usually have lower accuracy. For example, the WiFi RSSI based solutions use the existing WLAN infrastructure for both communication and positioning so they can be deployed easily at low cost, but their error is about 3-5 meters. The recently emerged hybrid indoor positioning systems can use multiple technologies so they can be more accurate and reliable, and they have the benefits and drawbacks of the selected technologies. For example, Baniukevic, Jensen, and Lu (2013) presented a system that uses both Bluetooth and WiFi technologies for positioning. Thus, the challenge in the development of indoor positioning system is to find a widely available, high accurate solution at low cost that can work in various environments with different technologies.

This paper presents the Indoor Localization and Navigation (ILONA) System which is a hybrid indoor positioning framework which provides localization, navigation and tracking functionalities to the users. Availability, flexibility, extendability were the key criteria during the design of the ILONA System. The existing systems fulfill some of these criteria but not all of them. For example, some solutions require specific hardware so they are not available for the masses. Moreover, most of the systems are built on an approach or method that cannot be changed or modified easily so these systems are not flexible and difficult to extend with positioning methods. ILONA System has no hardware requirement for the client devices and its component based design allows its extension with various positioning methods.

ILONA was designed to be easily available for users via smart phones. The system can handle the measurements of the most widely available sensors of smart phones, such as WiFi RSSI, magnetometer and Bluetooth. The major smart phone platforms were analysed during the design of the representation of the sensor data in the ILONA System. Client-server communication is performed via HTTP so it is platform independent. Although smart phones were the considered as the main client device, the systems can communicate with other kind of clients such as laptops or tablets. So the ILONA System provides easily available positioning and navigation functions to many kinds of client devices.

Localization, navigation and tracking functions are designed to be exchangeable, which makes the ILONA System flexible. Developers can define their own methods easily with the implementation of the corresponding interfaces that define requirements about the algorithm. The addition of custom algorithms allows the developers to tailor the ILONA System to the current needs of the environment and the project. The accuracy of the system could be increased with the customization of the positioning algorithm. Thus, for example, the ILONA System can use either triangulation or fingerprinting based method for positioning. Hence, the design and architecture of the ILONA System makes it flexible and eases its extension with novel positioning methods.

The ILONA System is a framework that provides the core functionalities of

indoor localization and navigation systems. Most of the currently popular and widely available technologies can be used for positioning in the ILONA System. These technologies can be used to create novel positioning methods for the ILONA System. This possibility of customization makes the ILONA System flexible and expandable. Thus, the presented architecture and components assist the development of indoor positioning solutions.

The rest of the paper is organized as following. Section 2 gives a brief overview of the existing indoor positioning systems and the technologies used. Functionalities of the ILONA System are summarized in Section 3. Section 4 presents the architecture of the ILONA System and the structure of its components. Experimental results and technological details are detailed in Section 5. The conclusion and directions of future work are summed up in Section 6.

2. Related Works

Indoor Positioning Systems have interested both academics and industrial firms since the early 1990s. Since then, various technologies have been used for indoor positioning such as infrared, ultrasound, RFID and WiFi. These systems are often categorized by the used technology, the required infrastructure or the estimation technique. Liu et al. (2007) and Deak, Curran, and Condell (2012) give a comparison of the existing systems and technologies.

The Active Badges system was the pioneer in indoor positioning. It was developed by Want and Hopper (1992) and based on infrared technology. The transmitters were carried by the people to be located and the receivers were installed in the building. Line of sight requirement of the transmitter and the receiver limited the applicability of the Active Badges system. Ward, Jones, and Hopper (1997) presented the Active Bats system, which used ultrasound to locate objects. The transmitters were built into the ceiling and the receivers were carried. The high installation cost limits the usage of the Active Bats system.

In the last decade, the RFID and WiFi based techniques have become popular. Ni et al. (2004) and Han and Cho (2010) presented the LANDMARC system, which is a RFID based system that uses active RFID tags. The tags were organized into a grid and placed in the floor. The receiver was carried by the located object. The LANDMARC system is cheap to install and the total cost depends on the number of the localized objects. The LANDMARC system can achieve an accuracy of less than 3 meters depending on the tag density.

RADAR was introduced by Bahl and Padmanabhan (2000). It uses WiFi Received Signal Strength Information for positioning. It is based on the fingerprinting approach so it has off-line and on-line phases. The WiFi RSSI is collected and stored in a fingerprinting database in the off-line phase. The system can be used for localization in the on-line phase. RADAR system estimates the current position based on the difference between the current RSSI measurement and the stored ones.

There are numerous fingerprinting based solutions in the literature. These systems mostly differ in the applied heuristics. Youssef and Agrawala (2005) presented the Horus system, which was based on a Bayesian approach. The dynamical changes of the measured RSSI values were also analysed during the work on the Horus system. The accuracy of the Horus system was achieved by the application of a client side filtering method which was presented by Youssef and Agrawala (2004).

WALRUS is a room–level indoor positioning system developed by Borriello et al. (2005). Ultrasound signals and a WLAN network are used to locate the mobile device. The ultrasound signals are emitted by PCs installed in each room. Position

and sound signal information is transferred via WLAN. There is no special requirement for mobile clients in the case of WALRUS system. The installation cost of the WALRUS system is lower than the Active Bats but it requires the installation of a PC per room.

WILL is a logical indoor localization system developed by Wu et al. (2013). WILL is based on the fingerprinting approach. It uses WiFi RSS and accelerometer information for positioning. In the training phase, it uses various data mining techniques to determine virtual rooms based on the measurements. This process makes the training phase faster and easier. The WILL system could achieve above 80% accuracy.

The above reviewed systems were designed and developed to estimate either coordinates or room-level position and these systems use a single technology and method. These requirements of these system limits their applicability. The ILONA System presented in this paper can overcome these limitations. Firstly, the ILONA System can be used to determine both coordinate and room-level positions. So, the ILONA System can be used to determine the room or floor where the users is located or estimate the user's location as coordinates. Secondly, the ILONA System can handle measurement from a wide range of sensors that are commonly available in modern smart phones, i.e. it is a hybrid indoor positioning system. Thus, ILONA System is not limited to a single technology. Finally, the design of the ILONA System allows its extension to different positioning methods and the already implemented methods can be changed easily.

3. Functionalities

ILONA System was designed with two major goals. Firstly, it should be a robust, high–performance and scalable indoor positioning system that can operate with any kind of client device. In other words, it has to be easily available and stable. Secondly, it is built to provide a common environment for implementation and comparison of various indoor positioning algorithms, which means flexibility and extendability.

3.1 *Use–Case*

The ILONA System distinguishes the common users from the administrator, which is shown in Figure 1.

The ILONA System provides `positioning` and `navigation` functions for the users. Users can use the system via their smart phones. The core function of the ILONA System is the `positioning`. The users can use their smart phones to acquire the sensor data and send it to the server-side component of the ILONA System that estimates their position. Hence, the users can determine their positions in an indoor environment. The `navigation` function can be used to find a way between two indoor locations. The user has to define the destination and the source location can be set by either the user or the `positioning` service. The `way finding` function is a part of the `navigation`. The ILONA System can be extended by various `way finding` algorithms. This function allows the users to find their destination in unknown places and the customization of the `way finding` algorithm can allow the generation of different routes for different users. For example, a patient's destination is Examination Room 101 and the shortest path leads up the stairs. But if the patient cannot climb stairs, then the System could generate another route

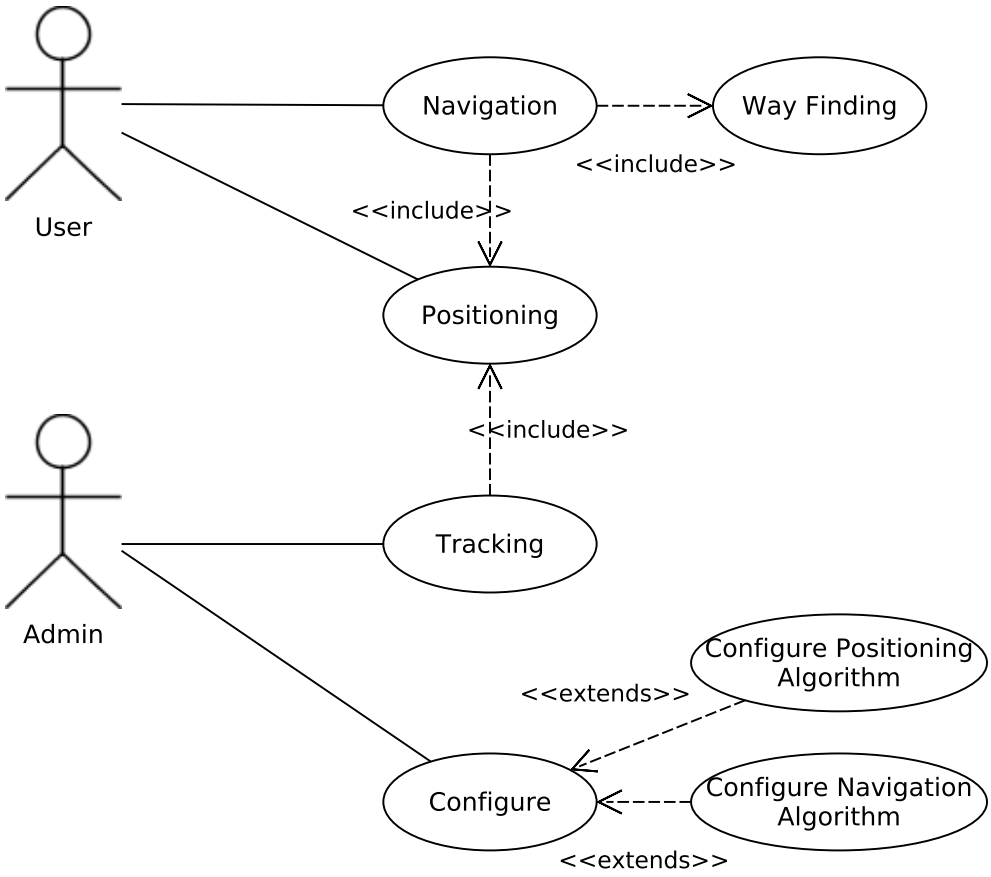

Figure 1. Use–Cases of ILONA System

that skips the stairs.

The administrator can use the `configuration` and `tracking` functions via the web interface of the ILONA System. The `tracking` function can be used to monitor the movement of the users. The estimated position of the tracked users are stored by the System, so the `tracing` function uses the `positioning` function. For example, the phones of the workers in a factory are tracked and the administrator can query the position of an employee who does not answer calls. The administrator can configure the navigation and positioning methods, which simplifies the testing of the different algorithms and their tuning. However, each method has a unique configuration page and the administrator needs knowledge about the algorithms.

3.2 *System Architecture*

Figure 2 shows the some main functions of ILONA System. The system is based on N–tier architecture and the client, server and database tiers are separated. This architecture was chosen due to its flexibility and scalability. The main responsibilities and challenges of these tiers are briefly reviewed.

Mobile Clients are responsible for data acquisition, conversion and communication with the server. Smart phones usually have heterogeneous sensor set the measurement depends on the available sensors of the given device. The measurements are converted into the common format which was presented by Tóth et al. (2015). Clients can be used in both on–line and off-line phases. In off-line phase,

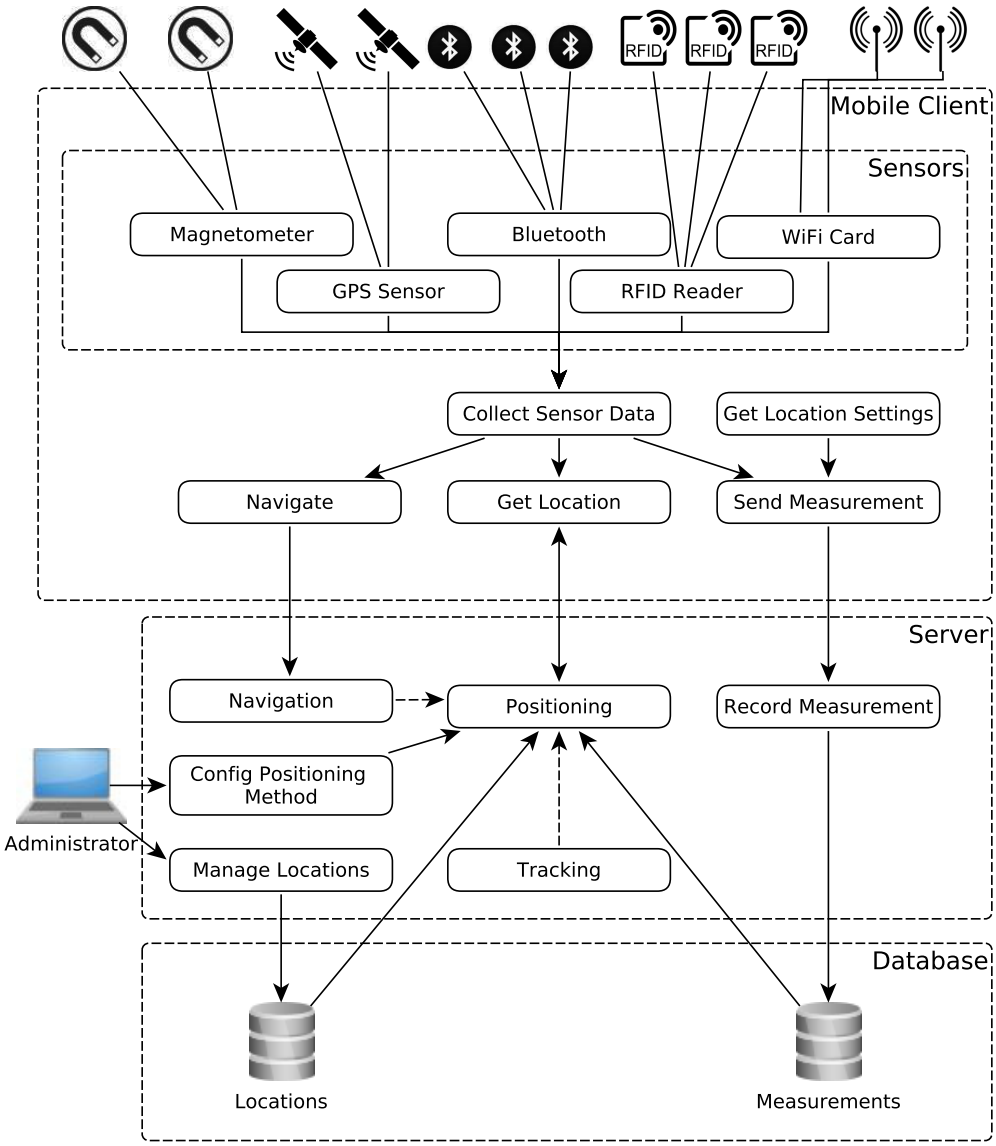

Figure 2. Functional Overview of the ILONA System

the user can set up the position. In on-line phase, the client is used for positioning or navigation. The implementation of the client application is challenging due to the huge number of available devices and the variety of the platforms and sensor sets. Moreover, the energy consumption of the application also could be an issue.

The server tier implements the business logic and integrates the functionalities which are assigned to components. The computationally intensive and costly algorithms, such as positioning, are implemented server side, which is beneficial for the following reasons. Firstly, it allows use of the client with different server instances. Secondly, the system is centralized; thus, maintenance and configuration of the clients is easy. Finally, the server side is scalable and clients' batteries are spared. These functionalities can be monitored and configured via a web application. The loose coupling of the components eases the changes of the different algorithms and makes the system flexible. The major challenge on the server side will be the guarantee of high accuracy with fast response in the positioning service.

Database tier is reliable for persistence storage. Separation of business logic and

data storage allows each component to have its own data format and database, which can run on different machines. Although the separation of the persistence tier is good for the developers, it may cause performance issues.

4. Components

The functionalities of the ILONA System are assigned to components whose development is independent. The components are based on the same module structure that eases the development. The modules are the smallest development units.

Figure 3 shows the components of the ILONA System. The web tier of the ILONA System integrates these components and wraps their functionalities, which is beneficial due to the following reasons. Firstly, the components can be developed, compiled and tested without the web application, which allows the development of reusable components. Secondly, the web application is reliable for the integration of these components, which eases and speeds up its implementation. Thirdly, the web application can integrate the required components; only thus can it be tailored to the needs of the users. It is essential for loosely coupled components to be able communicate via a standardized way, such as Java Messaging Service. The other components provide their functions as web services that are defined and implemented in the given component but cannot be deployed to a web server directly. The web tier uses these components and is reliable for the instantiation of their dependencies providing a context for their web services. Moreover, only web applications of this tier can be deployed to the server due to technical issues. The Spring framework was used to implement the web tier.

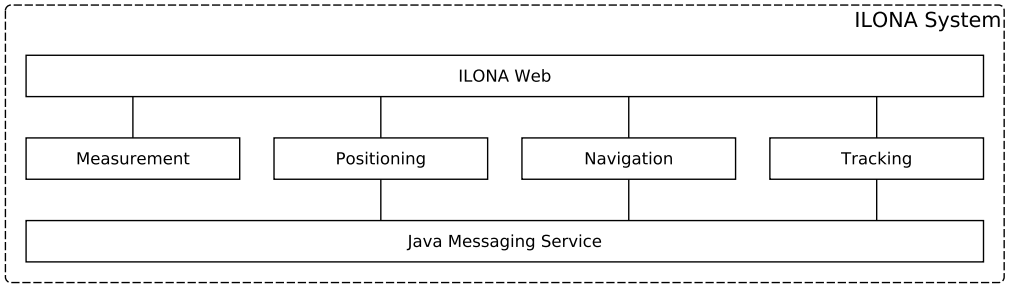

Figure 3. Components of the ILONA System

The ILONA system is built from four well-defined components, which are *measurement*, *positioning*, *navigation* and *tracking*. The functionalities are assigned to one of these components. There are dependencies between these components because a component can use the services of the others. In spite of these dependencies the ILONA System is scalable and robust due to its architecture and design. If the components communicate via HTTP, then they can be deployed on different application servers, which makes the system scalable. For example, the *positioning* component performs computationally intensive calculations so it should be deployed on a separate application server, while the *measurement*, *tracking* and *navigation* components are deployed on the same application server. Thus, their services will be available even if the server of that runs the *positioning* component has a high load. So the ILONA System is also robust.

4.1 *Measurement*

The *measurement* component is responsible for the storage and management of the measurements and zones. Zones represents the symbolic positions, which can be a room, a hall or even the floor of a building. Definition of zones depends on the user. The *measurement* component provides a web interface to record, list and delete measurements. The mobile clients can query the zones in JSON format via HTTP. The user can set up his own position, perform the data acquisition and then send the data to the server in JSON format for storage. The stored measurements can be queried and deleted via a web interface.

4.2 *Positioning*

The *positioning* components implement the localization algorithm. This component provides a web interface to invoke the currently used positioning algorithm. The client sends the measurements and the system estimates the position. Separation of the *measurement* and *positioning* components allows their independent development and simplifies the change of the positioning algorithm.

The *positioning* component defines an interface for the methods that can be used for position estimation. The interface was designed based on the Strategy design pattern that was detailed by Gamma et al. (1995) so the concrete methods have to implement these interface only and can be easily added to the ILONA System. Thus, the ILONA System can be easily extended with new positioning algorithms. Moreover, it provides a common software environment for the comparison of the different positioning methods. For example, a k-Nearest Neighbour and a Naive Bayes classifier based positioning method can be added to the system. The only requirement is to implement the positioning strategy interface defined by the ILONA System. Furthermore, these algorithms can be evaluated and compared in the same environment.

4.3 *Navigation*

Navigation in an indoor environment is challenging even for people. The *navigation* component of the ILONA System provides functions to the users to generate routes between the source and the destination. The source location can be set by the user or determined by the *positioning* component. The destination is chosen by the user. The route is generated by the way finding algorithm that can be configured by the administrator.

Existing indoor navigation methods are based on a map or an ontology. The FootPath is a map based indoor navigation system which was presented by Link et al. (2011). The Ontonav was one of the first ontologies for indoor navigation and was presented by Anagnostopoulos, Tsetsos, and Kikiras (2005). Ontonav was extended with restriction; thus it could generate different routes for people who lives with disabilities. This extension is called Onalin and was presented by Dudas, Ghafourian, and Karimi (2009).

The way finding algorithm of the *navigation* component is defined by an interface such as the positioning algorithm for the *positioning* component. Thus, the ILONA System can be easily extended with various way finding methods. In the current version of the ILONA System, an ontology based way finding algorithm was developed that allows navigation between rooms.

4.4 *Tracking*

The *tracking* component provides functionalities via the web interface for the administrator and is hidden from the mobile users. Identification and authentication of the users is required for tracking, and a user can have multiple tracked devices. The administrator can query the most recent location of the tracked users and visualize their movement via the web interface. The position information of the tracked users is provided by the *positioning* component. The *tracking* component is transparent for the users and it is placed between the user and the *positioning* component. The users call the tracking function as if it were the positioning function because the request is forwarded to the positioning function. The tracking component logs the result of the `positioning` service, then forwards it to the user.

4.5 *Component Structure*

Each component has the same structure, which is shown in Figure 4. These components are the smallest development and packaging units in the system. SOLID principles were kept in mind during the design of the component structure. Each module has a small well–defined set of responsibilities. Moreover, the expected behaviour and its implementation is separated, which increases the flexibility of the component. For example, this separation allows the integration of various positioning methods into the system.

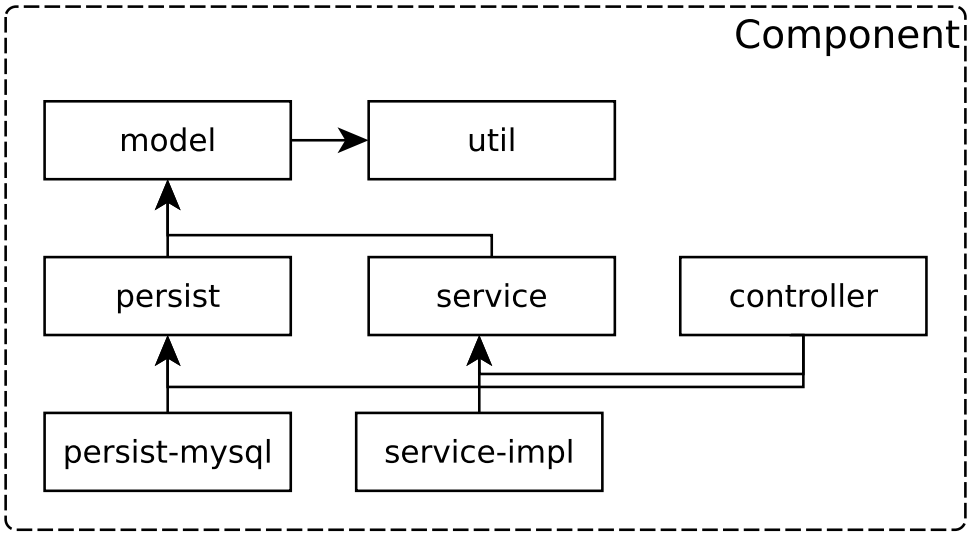

Figure 4. Components of ILONA System

The `util` module contains the so-called utility classes and functions, which cannot be assigned to any other modules. Logging and conversion typically belong to these functions. Every other module depends on `util` because it provides general functions.

The `model` module contains the business classes which are used by the other parts of the components. These classes are usually beans and POJOs. The basic domain specific behaviour, such as value validation, is also implemented in this module. Thus the module can also define domain specific exceptions. The `model` module is used by the `persist` and `service` modules.

The `persist` module defines interfaces and exceptions for storage and reading of business objects. Due to the principle of dependency inversion, the `persist` module defines only the expected functions and behaviour but it does not specify the implementation. The specific own exceptions of the `persist` module can be used to hide the implementation and technical details.

The `persist-impl` module implements the behaviour of the `persist` module with a specific technology. The separation of the definition and the implementation allows the creation of different implementations. Therefore, the system could support various database management systems easily. Current implementation stores the data in a MySQL database, chosen for its performance and availability.

The `service` module defines the functions of the components. The services perform operations on the model objects. The behaviour of the component is defined by interfaces and exceptions. The implementation of these services are placed in the `service-impl` module. Thus the implementation and the definition are assigned to different modules within the component. The separation of the `service` and the `service-impl` modules allows the interchangeability of the implementation and increases the flexibility extendability. For example, new positioning algorithms can be added to the system with the implementation of the corresponding interfaces and modification of the configuration files.

The `controller` module defines entry points for the various services. While the `controller` depends on the `service` and the `persist` modules, but it does not depend on their implementation. Due to the client–server architecture, HTTP communications and JSON encoding, server–side validation and type conversion is required. The `controller` classes perform the conversion of the client's input into the internal data model and vice versa. The errors and exceptions are also mapped to HTTP responses, which can be processed by the client application. Thus, the entry points and conversion functions are realized in the `controller` module.

5. Evaluation

ILONA System is under-development and extension but its architecture and core components are already implemented. Due to the extendability criterion of the ILONA System, its development cannot be considered complete. The ILONA System can be extended with novel positioning and way finding methods. On the other hand, the already implemented components of the system can demonstrate how the system fulfils its initial requirement of being easily available, robust, flexible and extendable.

The first goal was the usability; the ILONA System should be robust, scalable and available. Robustness and scalability of the system are closely related and they are ensured by the architecture. The loose coupling of the client and server side components and the RESTful interface simplifies the implementation of client side application so the ILONA system can be available for a wide range of mobile clients.

The second major goal of the development was to provide a common framework to implement, test and compare different indoor positioning methods. Thus, flexibility and extendability were set as main criteria of the system. The interface based design, object oriented design patterns and the hybrid data model together make the ILONA System flexible and extendable. The fulfilment of these criteria are detailed below from the point of view of implementation and application.

5.1 *Implementation*

ILONA System is a Spring-based web application implemented in Java. Maven was used to define the components, managing the dependencies and the build process. The source code is available on GitHub[1] and the compiled artefacts are stored in a local Nexus server.

The unit, composite and integration tests were implemented in JUnit and EasyMock was used to separate the test cases. The tests were run locally by the developers and a Jenkins Continuous Integration Server was also used to run the tests.

5.1.1 Architecture

The ILONA System is made of loosely coupled components that are shown in Figure 3. These components can be developed independently, which makes the development easier and faster. Hence, the `measurement`, `positioning`, `navigation`, `tracking` and the `ilona` components of the ILONA System are separate development units. Although these components can be developed independently, they may use some function of other components, i.e. some component may depend on a given version of another component. For example, the `tracking` component requires the `positioning` component, but they are both developed independently. So the `tracking` component can use an older version of the `positioning` component while the `positioning` component is extended with novel algorithms.

The loose coupling of the major components of the ILONA System ensures its scalability and robustness. The components communicate via HTTP and use JSON message. The `measurement`, `positioning`, `navigation` and `tracking` components can be deployed separately. For example, the `positioning` component may be running computationally intensive algorithms. In this case, the `positioning` component could be deployed on a separate server while the other components are deployed on the same server. Moreover, when a server that provides a service breaks down, it has no effect on the other components that do not require that service. Hence, the loose coupling also can make the ILONA System robust. So the components of the ILONA System can be deployed on different servers to make the ILONA System both scalable and robust.

The loose coupling and the HTTP based communication of the components makes the ILONA System available. The `ilona` component is a web application that integrates the other components and provides a web interface for the administrators. Smart phones and tablets were considered as the most widely used client devices. The clients can communicate with the server via HTTP, which is available in almost all modern devices. The server-side controllers consume JSON messages, which are plain texts that can be easily marshalled and unmarshalled at both client and server side. Although an Android application was developed for demonstration purposes, the server-side controllers were also tested with composed JSON messages that were sent to the server with Postman.

5.1.2 Service Tier

Flexibility and extendability were kept in mind during the design of the service of the components of the ILONA System. The definition and the implementation and the usage of the services are separated into three modules so they are different development units. The dependencies among these modules are shown in Figure 3 and detailed in Figure 5. The `service` module contains the abstract definition of

[1] https://github.com/ZsoltToth/ilona.git

the services because it defines only interfaces and exceptions that describes the expected behaviour. The `service-impl` module contains the implementation of the corresponding `service` interfaces and the business logic defined. The `controller` objects depend on the service definition so they can use an arbitrary implementation of the service used. Hence, the service implementation can be changed and the ILONA System is flexible.

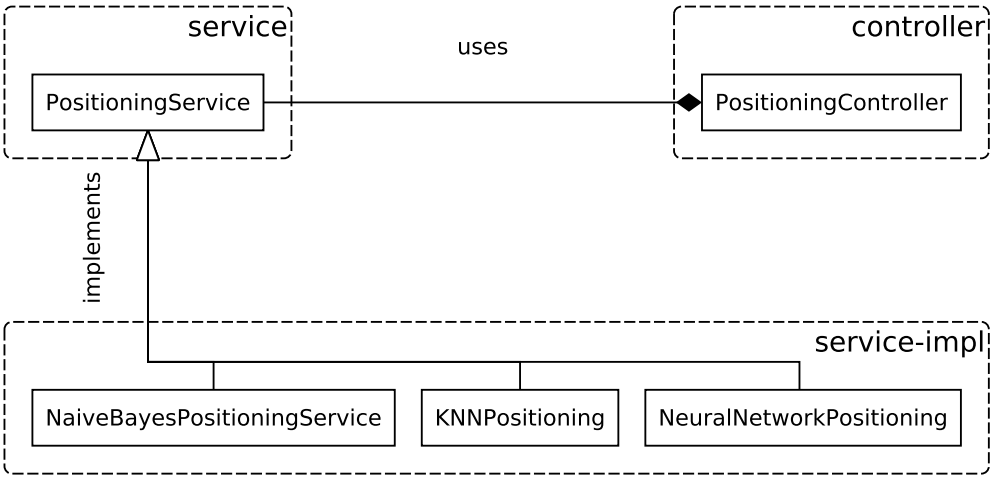

Figure 5. Separation of positioning service and implementation

Figure 5 shows the implementation of the `positioning-controller`, `positioning-service` and `positioning-service-impl` modules. The `PositioningController` is a Spring controller that contains methods mapped to specific URLs. The HTTP requests are converted and checked by these controller methods, then the method invokes the `positioning` function of the `PositioningService` object that is set for the controller. The `PositioningService` was created based on the Strategy design pattern presented by Gamma et al. (1995). Strategy design patterns can be used to change the behaviour of the object that uses the Strategy in runtime. The behaviour of the `PositioningController` can be changed in runtime and therefore the behaviour of the ILONA System can be changed in runtime. The `positioning-service-impl` module contains the implementations of the `PositioningService` interface. Other positioning methods can be added to the ILONA System by the extension of the `positioning-service-impl` module or addition of new a module to the system that contains the implementation of the `PositioningService`. The `navigation` component of the ILONA System was designed similarly so it can be also extended with novel way finding algorithms. Therefore, the ILONA System can be extended with Positioning and Navigation methods.

5.1.3 Persistence Tier

The flexibility of the ILONA System is also shown in the persistence tier. The `persist` modules of the components define the interfaces of the Data Access Objects. These interfaces usually contain the common CURD methods that can be used to store, query and modify business objects. The `persist` module do not contain any database management system specific information. Moreover, the `persist` module defines general exceptions that hide the technology specific exceptions. Hence, the ILONA System can be easily extended and used with various database management systems and solutions.

The `persit-mysql` module implements the interfaces of the `persist` modules. MySQL database management system was chosen for data storage because it is free, easy to configure and popular. The communication between the MySQL server and the ILONA System was implemented with the MyBatis persistence framework for the following reasons. Firstly, it is a currently popular persistence framework. Secondly, it does not hide the SQL scripts and can be easily configured with XML and Java Annotations. Finally, it provides classes to run SQL scripts that were used during the testing. Similar to the `persist-mysql` module, the ILONA System can be extended with different implementations that use other database management systems like PostgreSQL or Oracle.

5.1.4 Front-End Tier

Availability was a key criterion during the design of the front-end. An Android client was developed for testing which was able to perform measurements with different sensors and to communicate with the server via HTTP. The client converted the read sensor data into the format of the `model` of the `measurement` component. Thus, the server and the client used the same data format. Jackson JSON processor was used for marshalling on both client and server side. The HTTP based communication allows the implementation of client side application on other device that can use communicate via HTTP. Because smart phones, tablets and laptops can communicate via HTTP, the ILONA System is available to a wide range of clients and can be used with almost any kind of client.

5.2 *Experimental Results*

The ILONA System was deployed and tested in a real environment. The ILONA System was deployed into a virtualised environment and each sever-side component of the ILONA System was run on the same server. This experiment had three purposes. Firstly, the experiments allowed the testing of the ILONA System in a real-life scenario. Based on these test results, the exception handling and the value checking functions were refined. Secondly, the experiment was performed to demonstrate the usability and flexibility of the system developed. Experimental results of the demonstration suggest further directions for research on the performance and hardware requirements of the ILONA System. Finally, the data collected during the experiments can be used to extend and improve the ILONA System. This dataset can be used to run simulations that ease the evaluation and comparison of various indoor positioning algorithms.

5.2.1 Testing of the Server-side Components

ILONA System was deployed into a virtualised environment for testing and demonstration purposes. Testing and experiments allowed the refinement of the interfaces of web tier. Although most of the web Controllers were tested during the experiments, only two scenarios are presented below. The `measurement` and `positioning` components are used in these scenarios because the demonstration of the other components would require lot of technical details that are not strictly related to the ILONA System. For example, the presentation of the environment would be necessary to demonstrate the `navigation` component. Demonstration of the `tracking` component requires authentication and other security functions that are only used as 3rd party services in the ILONA System. Thus, the scenarios presented can demonstrate how the components of the ILONA System work together and can be used for positioning purpose.

The tests were performed with both Postman and an Android application. Post-

man is a Google Chrome plug-in that allows the creation, sending and monitoring of HTTP requests and responses. Postman was used because of the following reasons. Firstly, it allows the testing of the web services without the implementation of client side application. Secondly, the web services can been tested individually. Finally, the issues that could be caused by network are eliminated during these tests because both Postman and the ILONA Server can run on the same computer. Android client was developed in order to test the ILONA System in a real-life environment. The Android client communicates with the server via HTTP so it requires WLAN connection. The Android client was used to record measurements and test the positioning services.

The first scenario shows that the `Zone` objects can be queried from the server. The database stored the "bathroom" and the "kitchen" zones and the ILONA System was run on `localhost`. An HTTP GET request was sent to the `http://localhost:8080/ilona/resource/zones` URL that was processed by the dispatcher-servlet provided by the Spring Framework. The dispatcher-servlet mapped the HTTP request into the corresponding method of the `ZoneController`. The `ZoneService` object was called and it used the `ZoneDAO` to query the zones from the database. A collection of `Zone` objects yielded by the `ZoneController` was converted by the Jackson JSON Processor of the Spring Framework automatically. The JSON message sent by the server can be seen in Figure 6. The message was visualised by the Postman and it can be processed by client applications. The mobile client used this web service to set a combo box user interface element.

```
1 [
2 {"id": "9ff78a6a-2216-4f38-bfeb-5fa189b6421b",
3 "name": "bathroom"},
4 {"id": "d16a00e6-67c0-48ce-b75b-13417ecc0710",
5 "name": "kitchen"}
6 ]
```

Figure 6. List of Zones given back by the server

The second scenario demonstrates how can we use the positioning services via the web interface. The HTTP request sent contains a JSON message that represents a `Measurement` object and can be seen in Figure 7. Although the JSON message was composed manually in the current example, it can be easily created on any popular mobile platform. The HTTP request was sent to the `http://localhost:8080/ilona/getLocation` URL and was mapped to the corresponding method of the `PositioningController`. The JSON message expected should contain an id, a timestamp and the data from the sensors supported. The JSON message is converted into a `Measurement` Java object that is passed as a parameter of the corresponding method. The `PositioningController` verified the parameter and invoked the `PositioningService` to estimate the location. The positioning algorithm is realized as an implementation of the `PositioningService` interface and the `PositioningController` only forwards its result to the user. The `Position` object is converted into a JSON message and sent back to the user in a HTTP response.

Figure 8 shows the response JSON message sent by the server. Although the ILONA System can be extended with various positioning algorithms as it was shown above, in the current scenario, a dummy positioning service was used that always returns with the same `Position` object that represents the unknown position. The position estimated points to the $(0, 0, 0)$ coordinate and the zone called

```
{
"id": "f48439b3-8348-43c5-a5bd-d4e188b7dbeb",
"timestamp": 1482070088292,
"wifiRSSI": {
"rssiValues": {
"dummyAP1": -42,
"dummyAP2": -66
}
},
"magnetometer": null,
"bluetoothTags": null,
"gpsCoordinates": null,
"rfidtags": null
}
```

Figure 7. JSON Request to `PositioningController`

"Unknown". Android client was able to extract the zone and the coordinates and show them to the user on a pop-up message bubble.

```
{
"coordinate": {
        "x": 0, "y": 0, "z": 0
        },
"zone": {
        "id": "00000000-0000-0000-0000-000000000000",
        "name": "Unknown"
        },
"uuid": "2ee1f927-50af-45c9-ba44-f658f328135b"
}
```

Figure 8. JSON Response of `PositioningController`

These experiments show that the core components of the ILONA System works as expected. Although the scenarios presented only focus on the `measurement` and the `positioning` components, the other components were tested too. While a dummy positioning algorithm was used in the example, the ILONA System was tested with different positioning algorithms. The comparison of the various positioning methods is a possible research goal in the future. Tests with the Android client showed that the ILONA System can be used with the most popular mobile platform. The experiments also demonstrate the applicability and availability of the ILONA System.

5.2.2 Hybrid dataset

The Miskolc IIS dataset was recorded by the ILONA System and the dataset was presented by Tóth and Tamás (2016). The measurements were performed on the weekend in order to reduce the noise generated by the surrounding people. Only one type of client device was used for recording so that the diversity of the sensors would have no effect on the dataset. Each measurement contains data from Magnetometer, Bluetooth and WiFi sensors and has both an absolute and symbolic position that is defined by the user. GPS and RFID measurements were not performed because they were unavailable in the client device. More than 1500

records were recorded in the dataset and it covers approximately half of the entire building. The measurements were performed in a three-storey building and the dataset contains only the publicly available areas. The dataset can be downloaded from ResearchGate[1].

6. Conclusion

The presented ILONA System is capable of providing localization, navigation and tracking functionalities to a wide range of clients. The ILONA System consists of four major components: `measurement`, `positioning`, `navigation` and `tracking`. The core functionalities are assigned to well defined loosely coupled components, which makes the system flexible. Each major component of the ILONA System is an independent development unit that simplifies the development. Moreover, components can be deployed on different servers, which allows the scaling of the ILONA System and makes it robust. Availability, flexibility and extendability were the main criteria during the design.

The ILONA System was designed to be available for any kind of smart phone. HTTP support is the only requirement of the client because the ILONA System provides its services via HTTP. Client-Server communication is based on JSON messages that can be easily composed and processed in most mobile platforms. Hence, the functions of the ILONA System are available for most of the mobile platforms. The availability of the ILONA System was tested during the implementation and testing. Postman was used to test the web services during the development. An Android application was developed to test the ILONA System in real-life scenarios. Experimental results verified the availability of the ILONA System.

The flexibility criterion was considered during the design of the system architecture. Due to the loose coupling and the component based architecture, some components of the ILONA System can be deployed without the other components. The interface based development and the usage of the Strategy design patters allow the configuration of the `positioning` and `navigation` services even in runtime. The flexibility of the ILONA System is also shown in wide range of the sensors that can be modeled. ILONA System is a hybrid indoor positioning framework and was designed to handle data from the most commonly available sensors of the smart phones using these sensor data for positioning.

Extendability of the ILONA System was fulfilled by the interface based design and the high level of abstraction. Definition and implementation of the services are well-separated in the component structure used. Moreover, the services were designed based on the Strategy design pattern, which allows change in the behavior of an object in runtime. Thus, the ILONA System can be extended with novel positioning and way finding algorithms by the implementation of the corresponding service interfaces.

Experimental results showed that the ILONA System fulfill all of the initial criteria. Experiments performed were focused on the usability of the system. Performance analysis and stress tests will be performed in the future to determine the hardware requirements of the ILONA System. The dataset recorded during the experiments is being used for development, evaluation and comparison of various positioning methods. The ILONA System will be extended with novel methods based on the results of the simulations.

[1] 10.13140/RG.2.1.2829.6569

References

Anagnostopoulos, Christos, Vassileios Tsetsos, and Panayotis Kikiras. 2005. "OntoNav: A semantic indoor navigation system." In *1st Workshop on Semantics in Mobile Environments (SME05), Ayia,* Citeseer.

Bahl, Paramvir, and Venkata N. Padmanabhan. 2000. "RADAR: An in-building RF-based user location and tracking system." In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,* Vol. 2775–784. Ieee.

Baniukevic, Artur, Christian S. Jensen, and Hua Lu. 2013. "Hybrid indoor positioning with wi-fi and bluetooth: Architecture and performance." In *2013 IEEE 14th International Conference on Mobile Data Management,* Vol. 1207–216. IEEE.

Borriello, Gaetano, Alan Liu, Tony Offer, Christopher Palistrant, and Richard Sharp. 2005. "WALRUS: wireless acoustic location with room-level resolution using ultrasound." In *Proceedings of the 3rd international conference on Mobile systems, applications, and services,* 191–203. ACM.

Deak, Gabriel, Kevin Curran, and Joan Condell. 2012. "A survey of active and passive indoor localisation systems." *Computer Communications* 35 (16): 1939–1954.

Dudas, Patrick M., Mahsa Ghafourian, and Hassan A. Karimi. 2009. "ONALIN: Ontology and algorithm for indoor routing." In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on,* 720–725. IEEE.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Han, Kyuwon, and Sung Ho Cho. 2010. "Advanced LANDMARC with adaptive k-nearest algorithm for RFID location system." In *Proceedings of the 2nd IEEE International Conference on Network Infrastructure and Digital Content (ICNIDC'10), Sep 24–26, 2010, Beijing, China,* 595–598. IEEE Piscataway, NJ, USA.

Link, Jó Agila Bitsch, Paul Smith, Nicolai Viol, and Klaus Wehrle. 2011. "FootPath: Accurate map-based indoor navigation using smartphones.." In *IPIN,* 1–8. Citeseer.

Liu, Hui, Houshang Darabi, Pat Banerjee, and Jing Liu. 2007. "Survey of wireless indoor positioning techniques and systems." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 37 (6): 1067–1080.

Ni, Lionel M., Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. 2004. "LANDMARC: indoor location sensing using active RFID." *Wireless networks* 10 (6): 701–710.

Tóth, Zsolt, Péter László Magnucz, Richárd Németh, and Judit Tamás. 2015. "Data Model for Hybrid Indoor Positioning Systems." *Production Systems and Information Engineering* 7.

Tóth, Zsolt, and Judit Tamás. 2016. "Miskolc IIS Hybrid IPS: Dataset for Hybrid Indoor Positioning." In *Proceedings of the 26th Internatinal Confernece on Radioelektronika,* 408–412. IEEE.

Want, Roy, and Andy Hopper. 1992. "Active badges and personal interactive computing objects." *Consumer Electronics, IEEE Transactions on* 38 (1): 10–20.

Ward, Andy, Alan Jones, and Andy Hopper. 1997. "A new location technique for the active office." *Personal Communications, IEEE* 4 (5): 42–47.

Wu, Chenshu, Zheng Yang, Yunhao Liu, and Wei Xi. 2013. "WILL: Wireless indoor localization without site survey." *Parallel and Distributed Systems, IEEE Transactions on* 24 (4): 839–848.

Youssef, Moustafa, and Ashok Agrawala. 2004. "Handling samples correlation in the horus system." In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies,* Vol. 21023–1031. IEEE.

Youssef, Moustafa, and Ashok Agrawala. 2005. "The Horus WLAN location determination system." In *Proceedings of the 3rd international conference on Mobile systems, applications, and services,* 205–218. ACM.