Second term report

Project group 8

Aleksandar Rusinov, Dingzhong Weng, Jetnipat Sarawongsuth

Contents

1. Product Overview 3
2. Requirements background 4
3. Video of final product 6
4. Architectural diagrams 7
The Database schema 10
5. Development plan, iterations and forks in prototypes 19
5.1. Plan introduction and overview 19
5.1.1. Purpose, scope, and objectives 19
5.1.2. System overview, including system and software architecture 19
5.1.3. System requirements 19
5.2. Project team members resources breakdown by responsibility (management, software engineering, testing etc.) 19
5.3. Development (internal) processes, procedures, and work instructions 20

5.4. Plans for performing general software development activities 21
5.5. Software development processes 21
5.6. Software types/categories (i.e., operational software, test software 21
5.7. Plans for performing detailed software development activities 22
5.7.1. Software Development Planning 22
5.7.2. System Test Planning 22
5.8. Software configuration management 22
5.9. Other software development activities 22
5.10. Iterations made in the project development 23
6. Technical achievements, implementation details, use of design patterns 24
7. Management of the project including work packages completed between the team members 29
7.1. Requirements Management Plan 29
7.2. Schedule Control Plan 29
7.3. Management plan 29
7.4. Work packages 30
7.5. Configuration Management Plan 32
7.6. Testing and Quality Assurance Plan 32
8. Referenced materials cited and examples/trials made 33
9. Testing and evaluation of the product solution 34
9.1. Tests from hardware sensors 34
10. SDK/collaboration opportunities for further development 45
11. System manual and user manual 46
12. Community forum 47

1. PRODUCT OVERVIEW

Occupancy information for present, past and prediction for future plays important roles in many areas: smart buildings, control and optimization of Heating system, ventilation system, air conditioning (HVAC), Smarter and efficient company organization; Access control optimization, accident prevention and many others.

The Occupancy Engine is a cloud based system, intended to accurately quantify presence and movement of people in the rooms of a building based on a data sent from different sensors.

Main system features are:

- Format and logical validation, analysis and storing of all received data from different sensors
- Collected historical data is used for different queries and analyses – for example, calculation of current and past occupancy and movements for room, door and building
- Calculating the confidence level of counted occupancy
- Error handling module that performs automatic checks and correct errors procedures.
- Pattern of occupancy at a room and building level is estimated using historical data
- Users communicate with the system through regular browser. User Interface has the following features:
    - Consistent responsive design with login module and session support, help, contact information and forum section
    - Building plan, showing online current room occupancy of the building and colour differentiated crowdedness
    - Set of user parameterised tabular and graph reports

Making use of algorithms to analyze and estimate room occupancy in a building using sensor measurements from diverse sources and historical data. Information regarding occupancy levels and distribution patterns of movement in buildings may follow different patterns based on special events or times of the day. The room occupancy may vary seasonally, with time of the day or even days of the weeks and holidays. For example, a living room occupancy pattern is usually higher during the day than during the night.

2. REQUIREMENTS BACKGROUND

The key requirements to our system are:

- to accurately quantify the number of people in each room of a building
- to calculate the confidence level of the number of people in every room
- to store a historical data for further analyses
- to create an error handling module with automatic checking and correction of the errors
- to have user interface, showing:
    - current and past number of people in every room,
    - confidence level,
    - set of different reports

We completed all functionality as planned in our PoC and created some additional features during the process of development. The development plan followed our PoC. We successfully finished all 'must have' and 'should have' features and partially 'could have' and 'would like to have', with the opportunity to complete them in future development .

PRIORITIES	FEATURES	COMPLETED
MUST HAVE	• A system that is able to accurately quantify the number of people in each room in a building.	• Yes
	• A user interface that the users can interact with to check how many people are currently in a room and in the past. This will be in a form of a website.	• Yes
	• An error handling module that can cope with hardware detection errors. For example, the case where the hardware says that there are more people leaving the room than there actually are in the room.	• Yes
	• Storing the data for past occupancy.	• Yes
SHOULD HAVE	• A colour coding scheme for the floor plan map to represent how crowded a room is.	• Yes, and with an additional functionality – online connected to the database building plan to show current number of people

		in every room.
COULD HAVE	• A system that is able to predict, accurate up to a certain degree, the number of people that will possibly be in a room at a given time on a given day.	• Calculated weekly convex shows occupancy pattern for every weekday and could be used as a base for occupancy prediction
WOULD LIKE TO HAVE	• A system that uses machine learning to predict very accurately the number of people that will be in a room at a given point in the future. • A system that can, not only count the number of people in a room at the time as well as predicting the number of people at any given time in the future, but also identify the different people in the room.	• Historical data collected from sensors in snapshot tables and data from automatic error correction in error correction log table, along with patterns in configuration tables forcedrules and occupancylevel could serve as a input to inference machine, based on neural network/Markov chains for machine learning. • Additional types of sensors and methods should be implemented – beacon devices, face recognition video technology etc.

3. VIDEO OF FINAL PRODUCT

4. ARCHITECTURAL DIAGRAMS

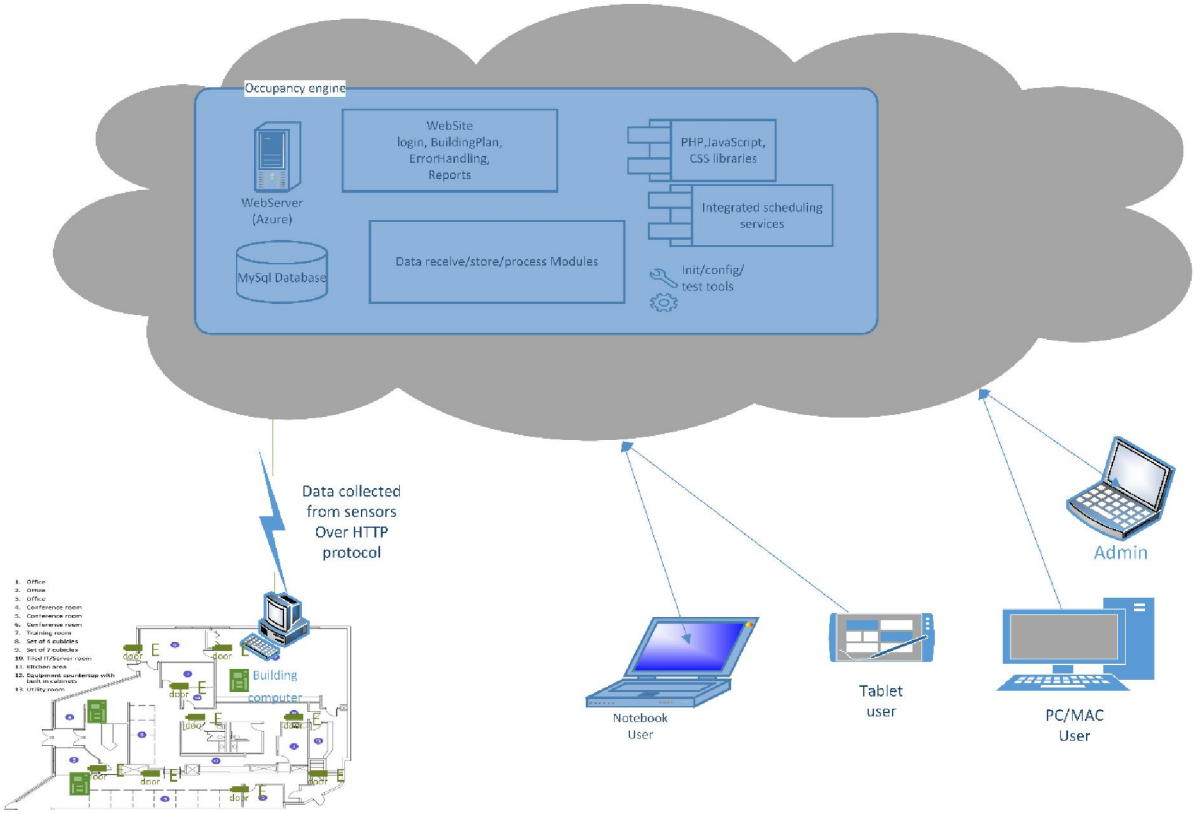


Figure 4-1 System overview schema

System architecture overview

Our product is a cloud based system:

- Server side – Microsoft Azure cloud platform
- MySQL database server
- PHP scheduled modules to receive, store and process input data.
- Scheduled procedures for errors handling and convex calculation
- HTTP transfer to collect data send from an Arduino controlled sensors

User interface using foundation framework responsive design, HTML5, CSS, JavaScript
Modules for:

- Initializing Java module - for set up, database creation, configuration and initial data loading in system parameter tables.
- Java Client side module - Simulator of sending data online from hardware sensors
- PHP module to create and load test samples of historical data

We use Github repository with version and source control from where the source code is deployed on Azure and local host staging environment for the product and its iterations.

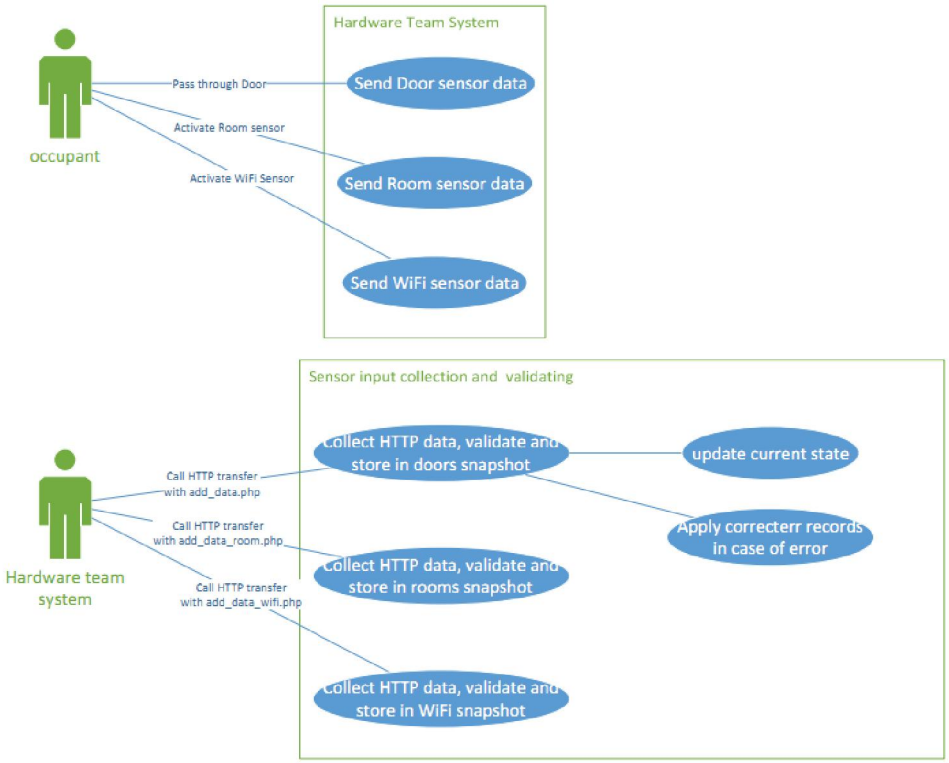


Figure 4-2 Use cases for sensor events and data transfer

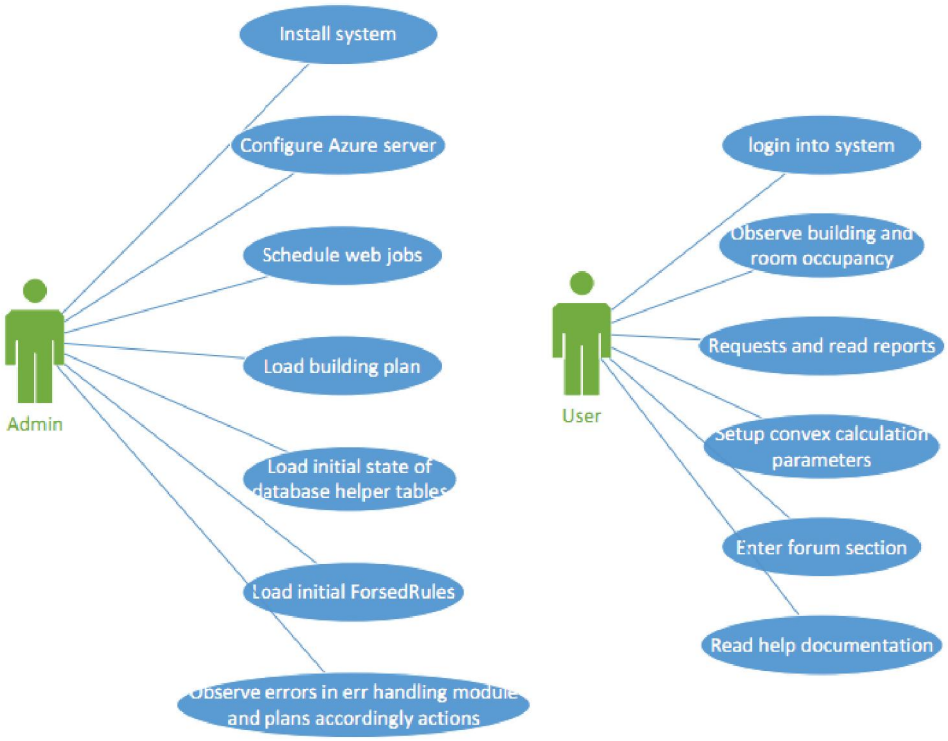


Figure 4-3 Use case for system configuration and user interaction

The Database schema

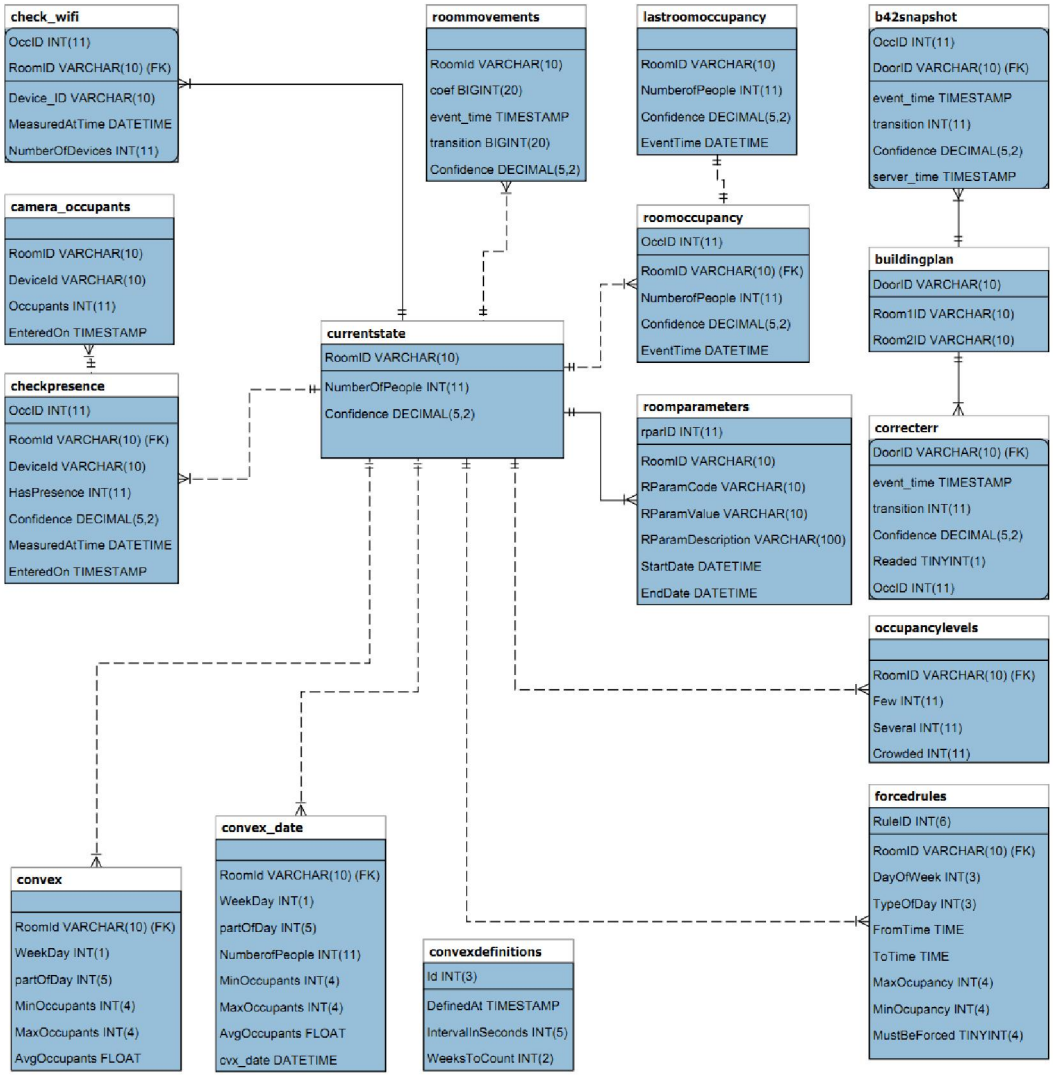


Figure 4-4 Database Entity-Relatonship schema

B42Snapshot stores the data that comes directly from the Arduino. It shows the transitions at a given door along with the confidence level, direction sign and timestamp of the event.

BuildingPlan lists all the doors in a building as well as their adjacent rooms. For this model we assume that a door is two sided. The transition is positive when it is from Room1 to Room2 and negative if it is from opposite direction.

CurrentState stores the number of people currently occupied a given room, along with a cumulative confidence level. This table is created to allow faster access to the current state of each room.

RoomOccupancy stores the past data for each room occupancy at a given moment. A parameter time interval is used, for example 1 day, 1 hour or 5 minutes depending on the people movement dynamic and the snapshot will be generated. This database will be updated during a period where there are no activities. For example, very late at night.

OccupancyLevels gives information about the crowdedness of each room. The values vary depending on the area of a room. The occupancy level is associated with a color. The values in Empty, Few, Several, Crowded present an upper bound for the level. Empty is always 0. For example if we have Few – 4, Several – 10, Crowded – 30, and we have green for Empty, yellow for Few, orange for several and red for Crowded and the system shows that there are 14 people in a room, it will have red label.

RoomParameters is a system table that specifies various characteristics about the rooms. For example, one can store HVAC (heating regulation and air-conditioning) conditions about every room. Since there systems use a lot of energy, a demand driven control will be essential in minimising the energy consumption. The value parameter represents the number of people that should be in the room for a certain activity to start. The dates show the period that this activity is valid.

CheckPresence stores the data that comes directly from the room sensors. It shows the presence (Yes/No) in a given room along with a device number, confidence and timestamp of measurement.

Check_WiFi stores the data that comes directly from the Wi-Fi sensors. It shows the number of devices in a given room along with a device number and timestamp of measurement.

CorrectErr stores a correction transitions of error handling module. Structure and fields are same as of b42snapshot.

ForcedRules is a table that specifies max and min people limits for the room at agiven period of a given day of week, typeofday (working or holiday), and flag for forcing rule. This is used by scheduled error correction procedure to enforce sets limit.

ConvexDefinitions stores the parameters used in building and populating convex and convex_date tables. These parameters are ‘length in weeks in the past’ and ‘interval length to divide days (in seconds)’.

Convex_date stores calculated history of a room’s occupancy in the defined weeks in the past in the form of min, max, average values per intervals of a days.

Convex stores min, max and the average values of the room's occupancy calculated per intervals of a weekday.

There are several helper views for facilitating queries and data processing procedures: **roommovements**, **camera_occupants** and **lastroomoccupancy**.

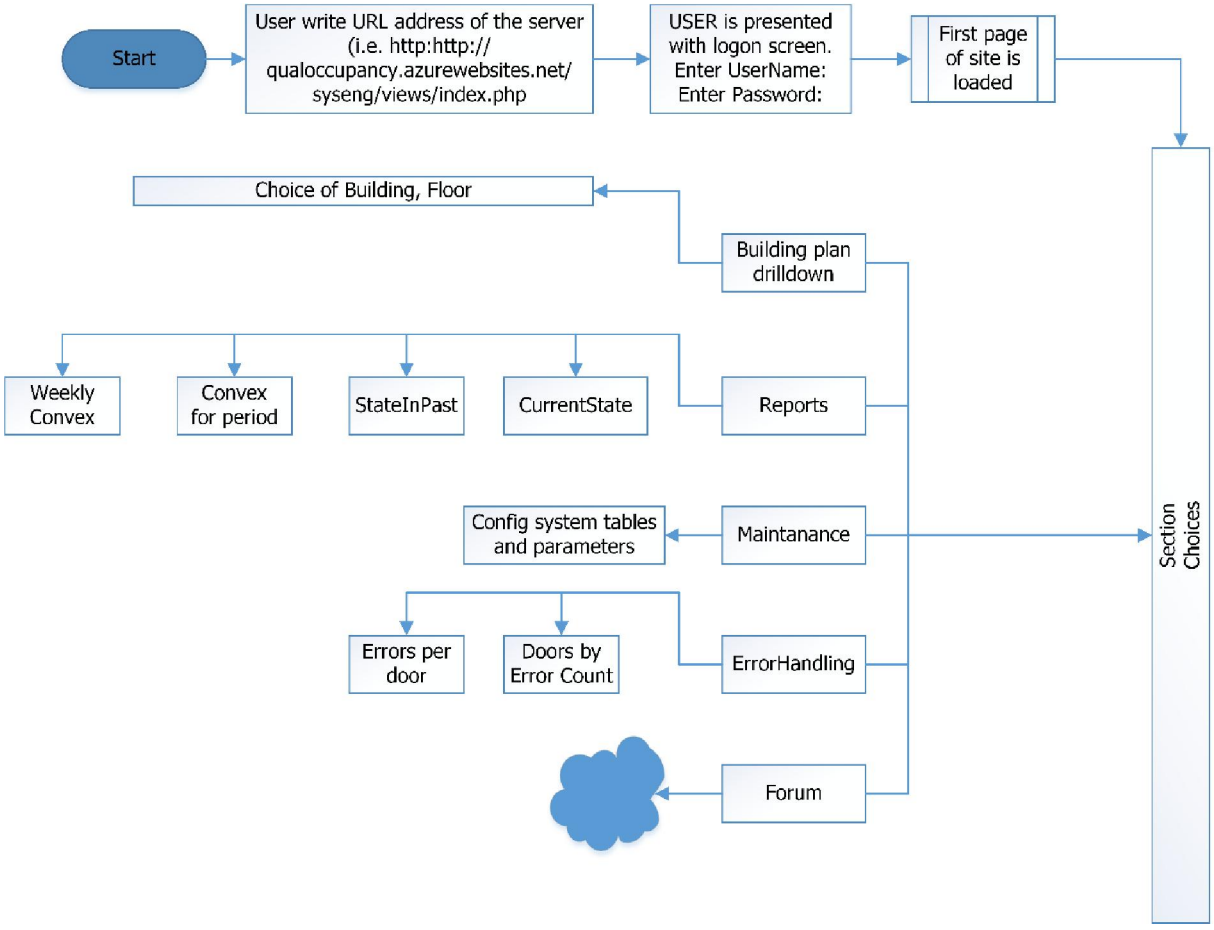


Figure 4-5 System use diagram

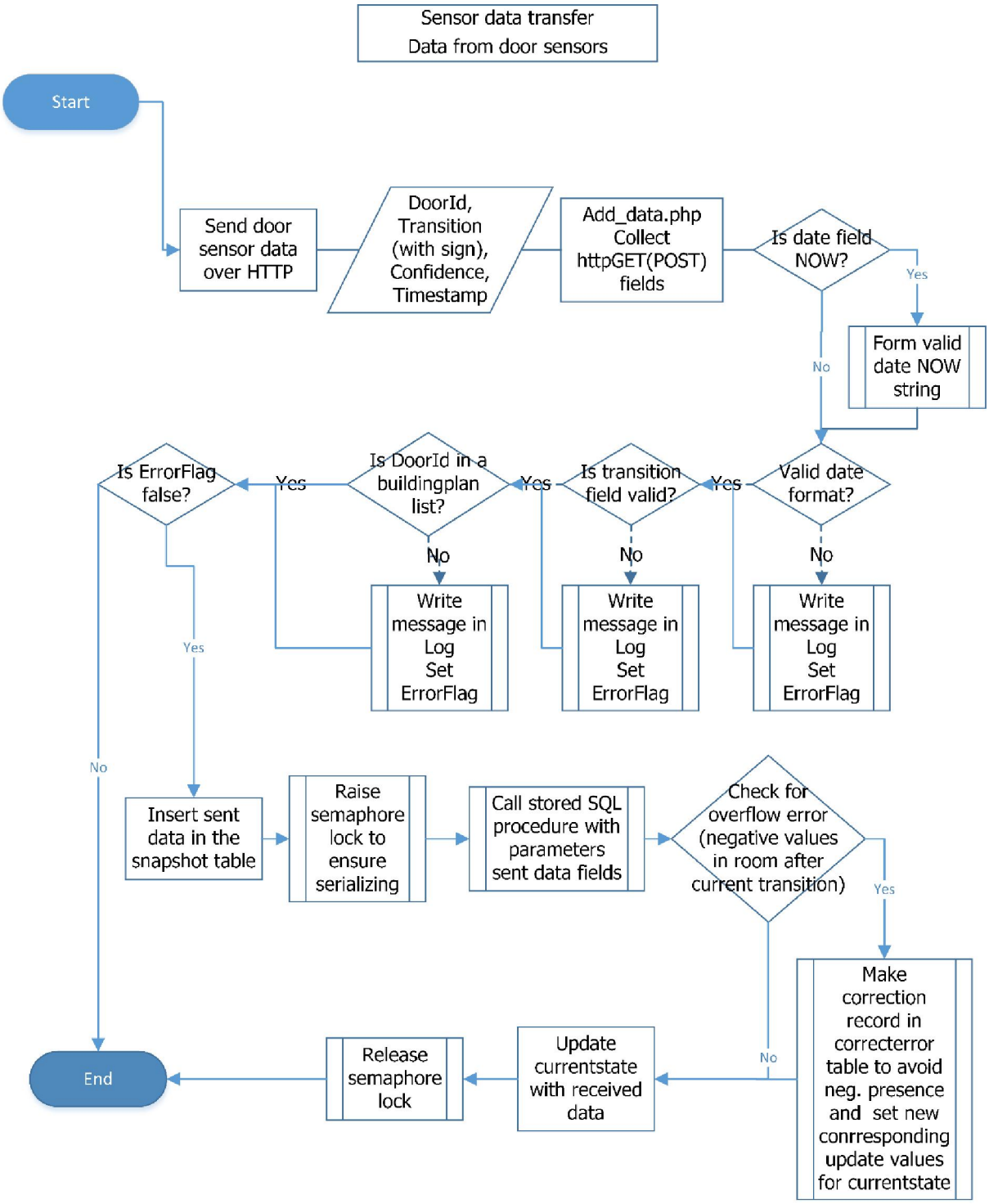


Figure 4-6 Data from door sensors transfer diagram

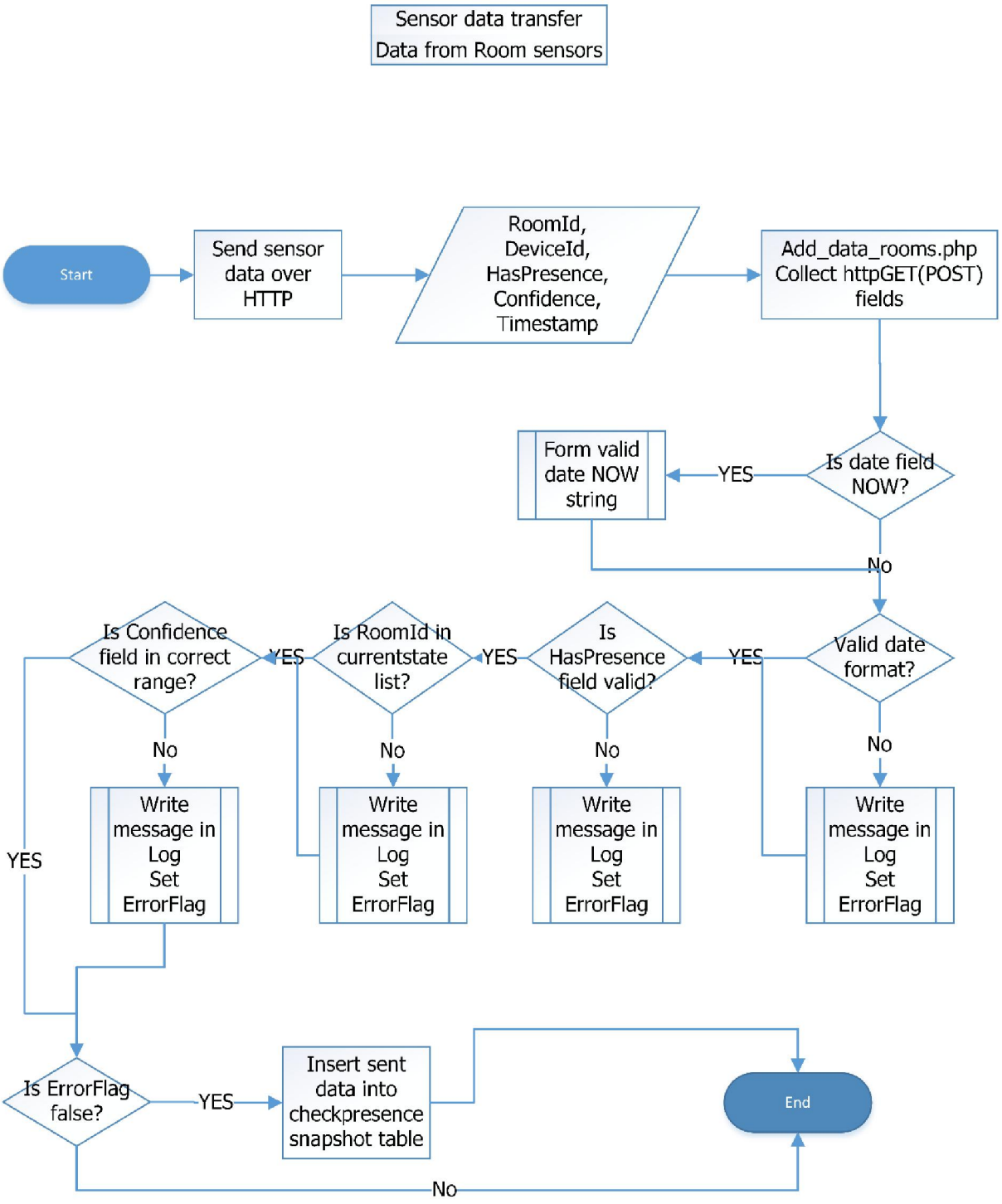


Figure 4-7 Data from room sensors transfer diagram

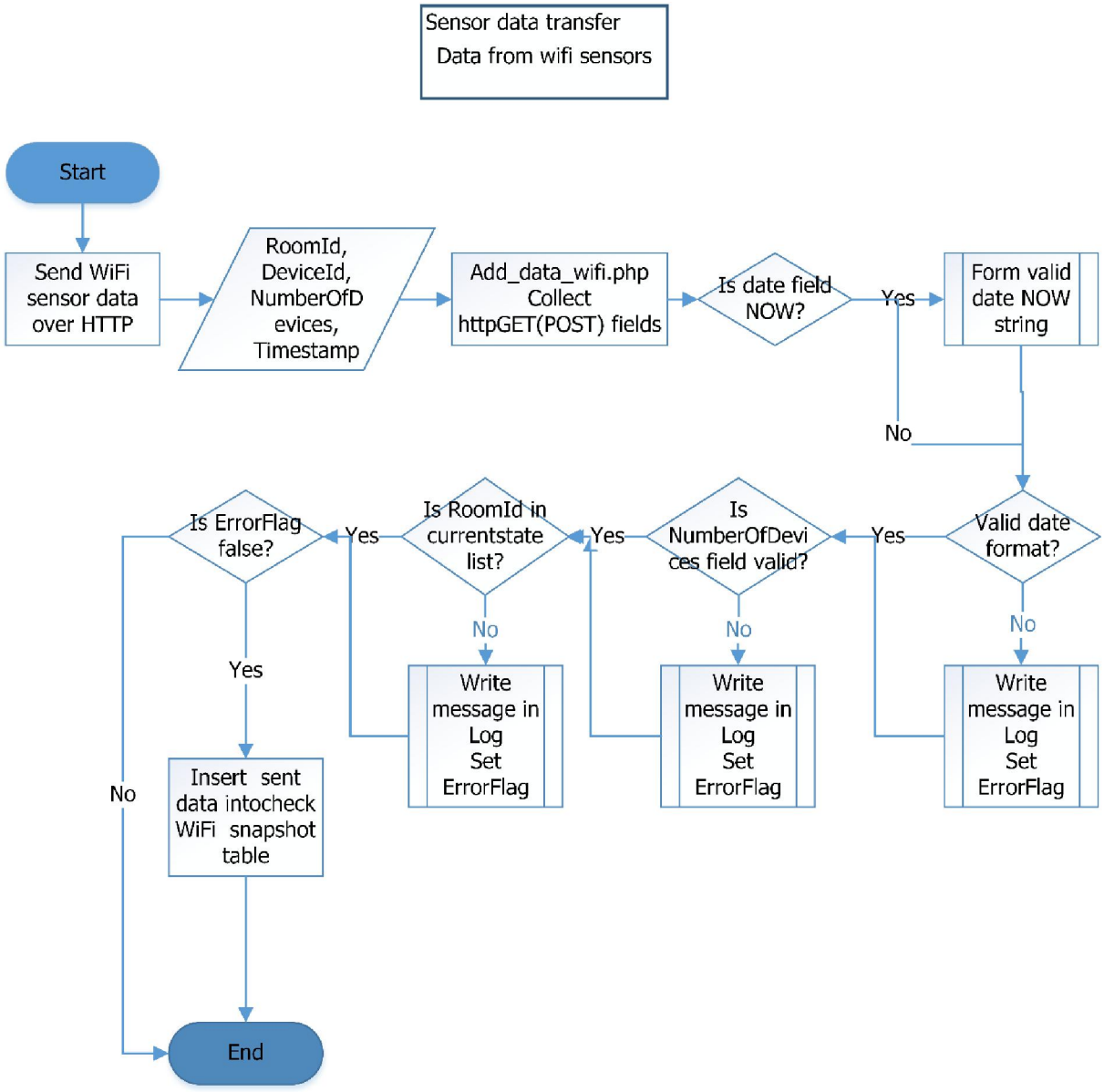


Figure 4-8 Data from Wi-Fi sensors transfer diagram

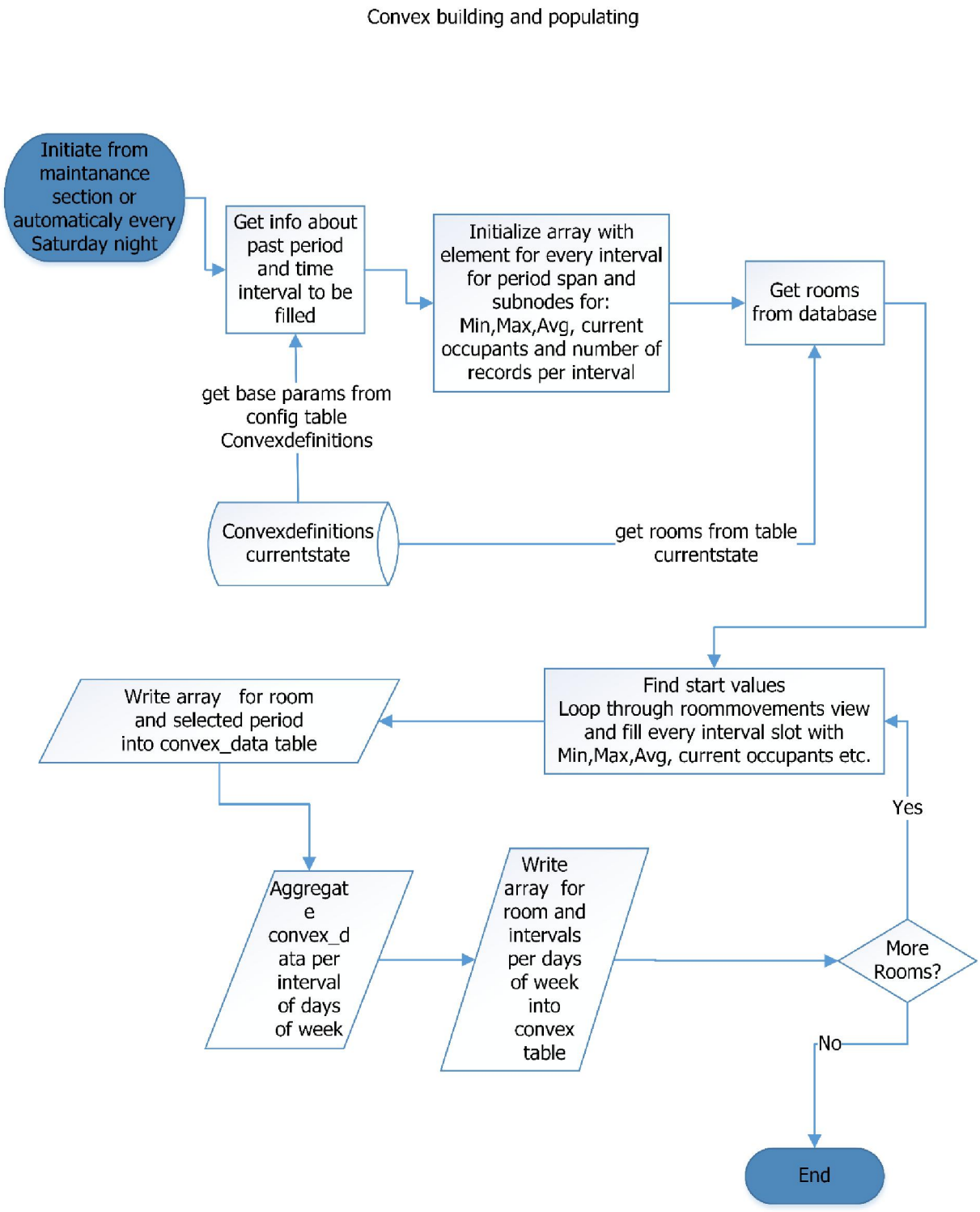


Figure 4-9 Convex and weekly convex calculating

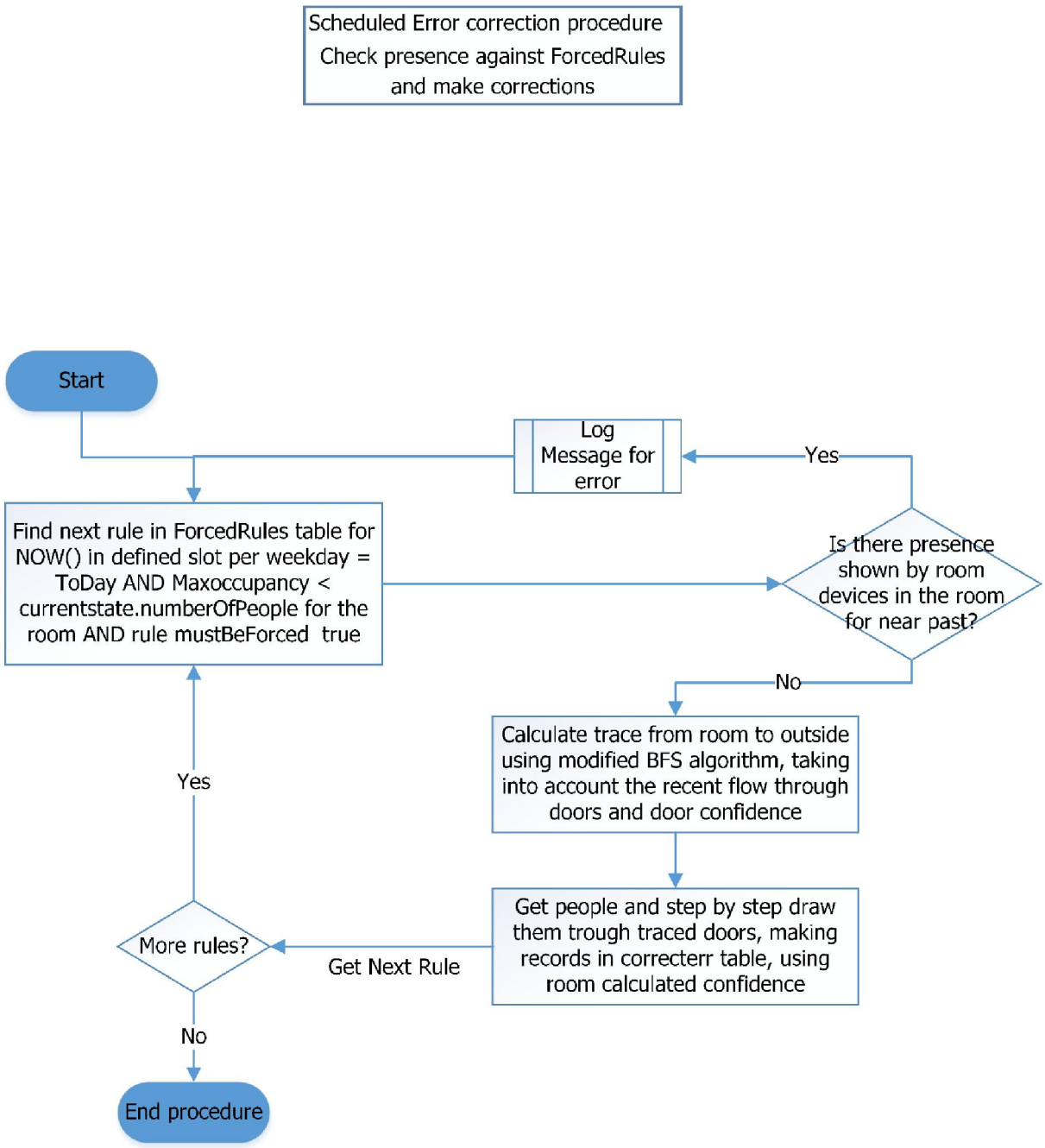


Figure 4-10 Process of error correction using predefined rules in forcedrules table

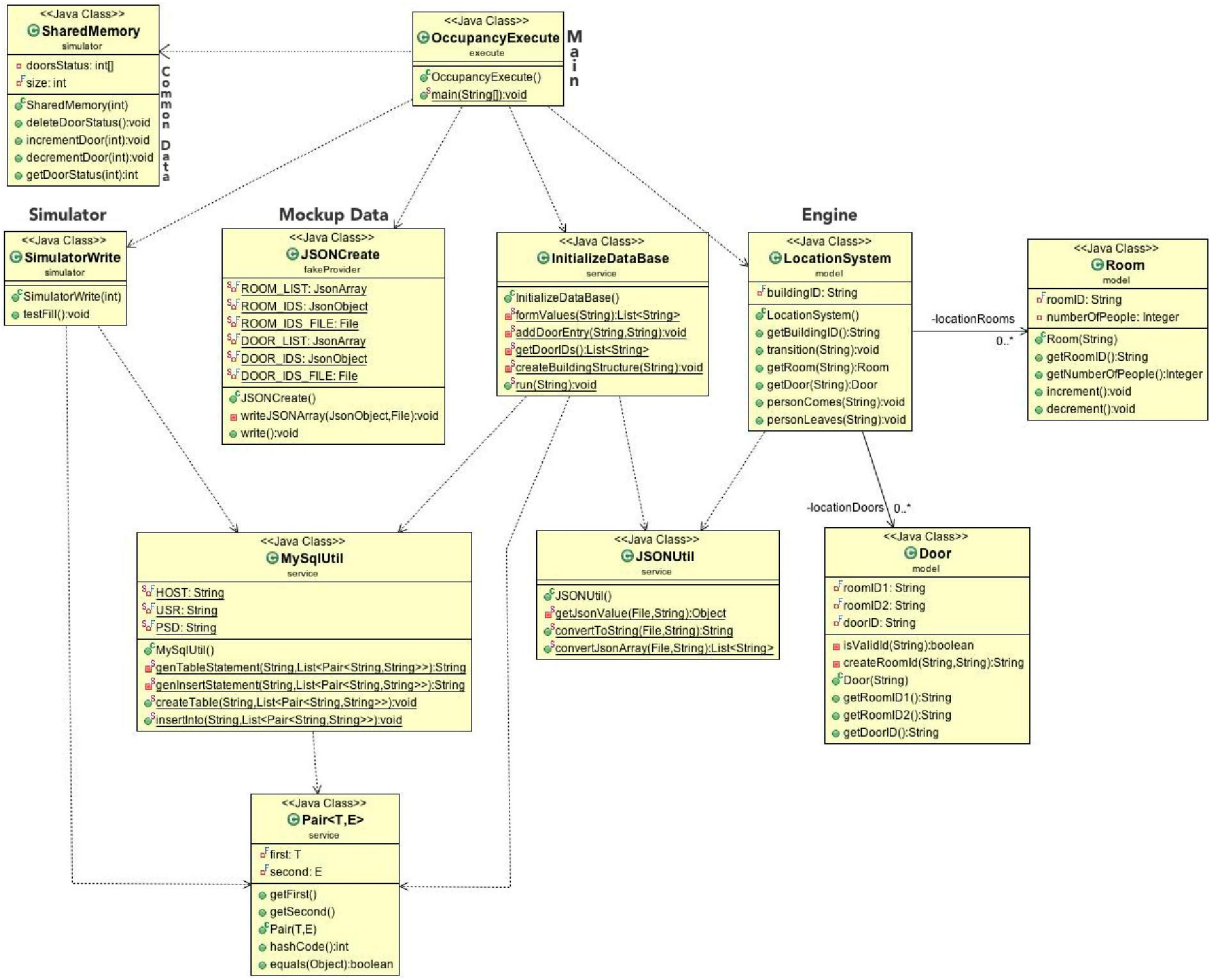


Figure 4-11 Java Database initialisation Module

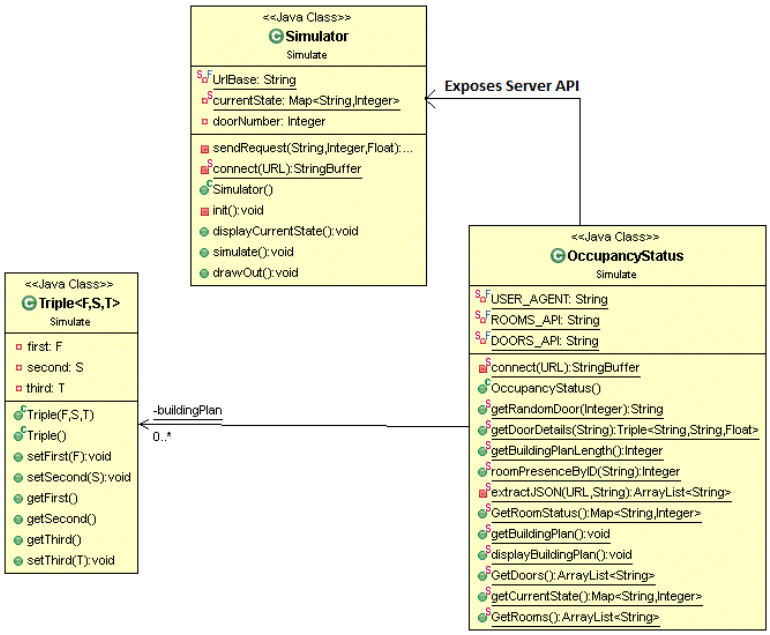


Figure 4-12 Java Simulator Module

5. DEVELOPMENT PLAN, ITERATIONS AND FORKS IN PROTOTYPES

5.1. Plan introduction and overview

5.1.1. Purpose, scope, and objectives

This Software Development Plan (SDP) establishes the plan for software building, testing, and deployment of the Qualcomm Occupancy Engine System (QOE). The QOE is being developed under the supervision of the Computer Science Department of UCL and under close contact with Qualcomm as a client. Updates to this SDP will address future QOE software upgrades.

5.1.2. System overview, including system and software architecture

In general, an Open source software approach is used including a description of the code, its specific origin, how it is controlled, how it is tested or analyzed.

The Qualcomm Occupancy Engine System involves the development and server deployment of the software modules of web based system:

- Server side – Microsoft Azure cloud platform
- MySQL database server
- PHP scheduled modules to receive, store and process input data.
- Scheduled procedures for errors handling and convex calculation
- HTTP transfer to collect data send from an Arduino controlled sensors

User interface using foundation framework responsive design, HTML5, CSS, and JavaScript.

Modules for:

- Initializing SQL database module with Java and JSON files - for set up, database creation, configuration and initial data loading in system parameter tables.
- Java Client side module - Simulator of sending data online from hardware sensors
- PHP module to create and load test samples of historical data

Qualcomm Occupancy Engine System is encapsulated and can be easily deployed on different platforms – cloud, Linux/Windows/Mac OS with different WEB servers/Relational Database management systems.

5.1.3. System requirements

The project requirements are described in Chapter 3

5.2. Project team members resources breakdown by responsibility (management, software engineering, testing etc.)

	Primary Role	Secondary Role	Tertiary Role
Aleksandar Rusinov	**Team leader:** Manages the delegation of the work in accordance to the skills of the team members. Responsible for group meetings and meeting deadlines.	**Client and Supervisor liaison:** Chief Client and Supervisor liaison, Makes connection with the client and the UCL supervisors and informs the team members of any changes.	**Main developer and product manager:** In charge of the software development and product management. Main coder and module builder. Designs how the system should act, what simulators should be done and gives ideas how to make the product more attractive.
Dingzhong Weng	**Head of Visual and webpage:** Has created the webpage of the project and also made the UML diagrams.	**Minutes and Report Redaction:** Vice Report manager. Has to deal and proof check reports and meetings when the head reporter is absent.	**Client Liaison:** Vice client and UCL supervisor liaison. Connects with client, UCL supervisor and third party advisor when the chief Liaison is absent
Jetnipat Sarawongsuth	**Chief Editor:** Looks after all reports, minutes of meetings and documentation. Reviews the reports and makes redaction.	**Head of Research:** Looking after all the researches. Studing different products, communication devices and way of connection with the arduino.	**Testing:** Provides means of testing. Gives solutions for simulator tests and code testing coverage.

5.3. Development (internal) processes, procedures, and work instructions

The QOE Project apply an Incremental (Preplanned Product Improvement) strategy with iterations loops to develop and evolve the functional capabilities of the system. The releases of development are created and applied to the GitHub repository and feedback is incorporated in next releases.

5.4. Plans for performing general software development activities

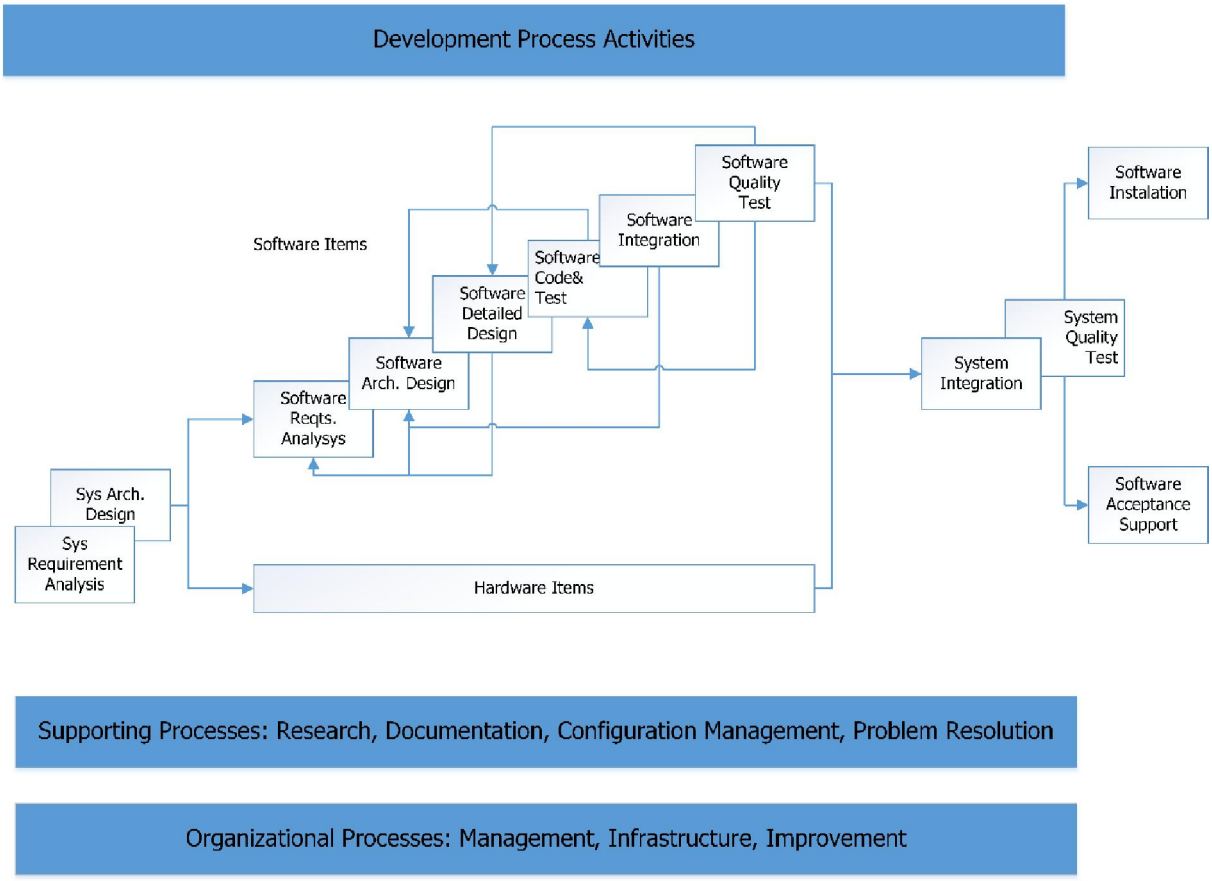


Figure 5-1 Development Process Activities diagram

5.5. Software development processes

The Incremental Life Cycle Model has been used to guide the content and format developing the SDP. The allocation of functional requirements for each build was negotiated with the client (Qualcomm) on weekly meetings and documented in the GitHub repository for each build. The SDP integrates reusable software from existing sources with newly-developed software. Software design and coding was performed using an object oriented design approach and generate class and object process interaction diagrams. Artifacts of software development activities was deposited into GitHub repository.

The Test Plan (TP) cases had been prepared and executed as described in the Chapter 9. Software errors uncovered during tests were had been addressed, providing analysis and repair as required.

5.6. Software types/categories (i.e., operational software, test software

Development environment consists of local desktop/laptop with installed WEB server (with PHP interpreter – i.e. Apache)/MySql database, Java development environment (i.e. Eclipse), internet access (to pull/push artefacts in GitHub repository), source editors and document editing software (i.e. MS Office).

Production environment is Microsoft Azure cloud platform.

For test environment, we use desktop/laptop with internet connection, standard browser (i.e. IE, Chrome, Mozilla latest version), installed PHP interpreter for Use case tests and Java runtime for the Simulator of sensor data.

5.7. Plans for performing detailed software development activities

5.7.1. Software Development Planning

The plan for all software development follows software engineering 'best' practices. Microsoft Project was used to develop and maintain the QOE Project master plan and schedule. The Project leader had used weekly project meetings to maintain the status of the software project and to resolve any conflicts or changes that might occur.

5.7.2. System Test Planning

The intent of system test planning is to validate that the system meets its performance requirements. It is the responsibility of the Test Manager to direct the development of system test plans and procedures, and conduct the system tests. Testing is performed to validate each component's ability to meet its stated requirements and to ensure interoperability of the major software components. System qualification testing is prepared to demonstrate to the client that system requirements have been met. It covers the system requirements agreed and written in the system requirements part.

5.8. Software configuration management

We have created script procedures for setting Azure environment, transferring system from GitHub repository and calling modules for the initial database creation and loading. Fully automatic installation and configuration is not provided in this release. Depending on target server system, different procedures are to be created in next releases.

5.9. Other software development activities

Here is technical and management approach to coordinating and providing oversight of software development activities undertaken by the QOE Project. Timely technical and management oversight at the appropriate level is necessary to accomplish the activities listed below:

- Track progress against plans
- Identify and resolve problems
- Activities and relationships associated with planning and conducting integration

Here is to be addressed other related development actions and schedules that could impact QOE Project.

5.10. Iterations made in the project development

Phase	Iteration	Description	Issues and Risks addressed
Requirement Phase	R1	Defines initial product requirements and Software Development Plan	Develops initial requirements documents for review
	R2	Defines product requirements and Software Development Plan	Develops realistic Software Development Plans and scope
Design phase	D1	Complete analysis and design for major use cases. Complete initial design of architecture	Architecture can be reviewed. High-risk use cases can be reviewed
	D2	Complete analysis and design for all use cases. Complete prototype of architecture	Architectural issues clarified Technical risks minimized
Software development phase	S1	Implement skeleton of architecture	Architecture available for implementors
	S2	Implement and test high-risk use cases	High-risk use cases are implemented
	S3 – Develop Alpha release	Implement and test low-risk use cases. Complete unit/procedure testing	Defects minimized
System testing and deployment phase	T1	System qualification testing and configuration/ deployment to production site	Qualcomm Occupancy Engine final product was released

6. TECHNICAL ACHIEVEMENTS, IMPLEMENTATION DETAILS, USE OF DESIGN PATTERNS

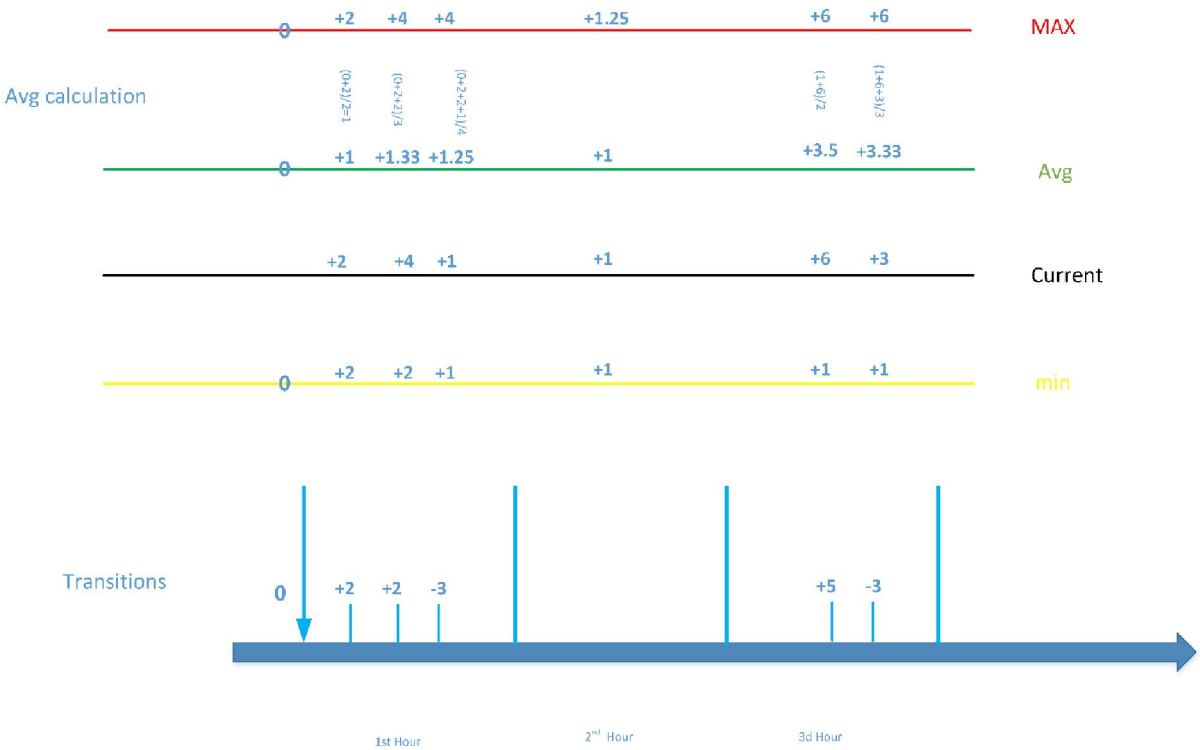
6.1. System functionalities:

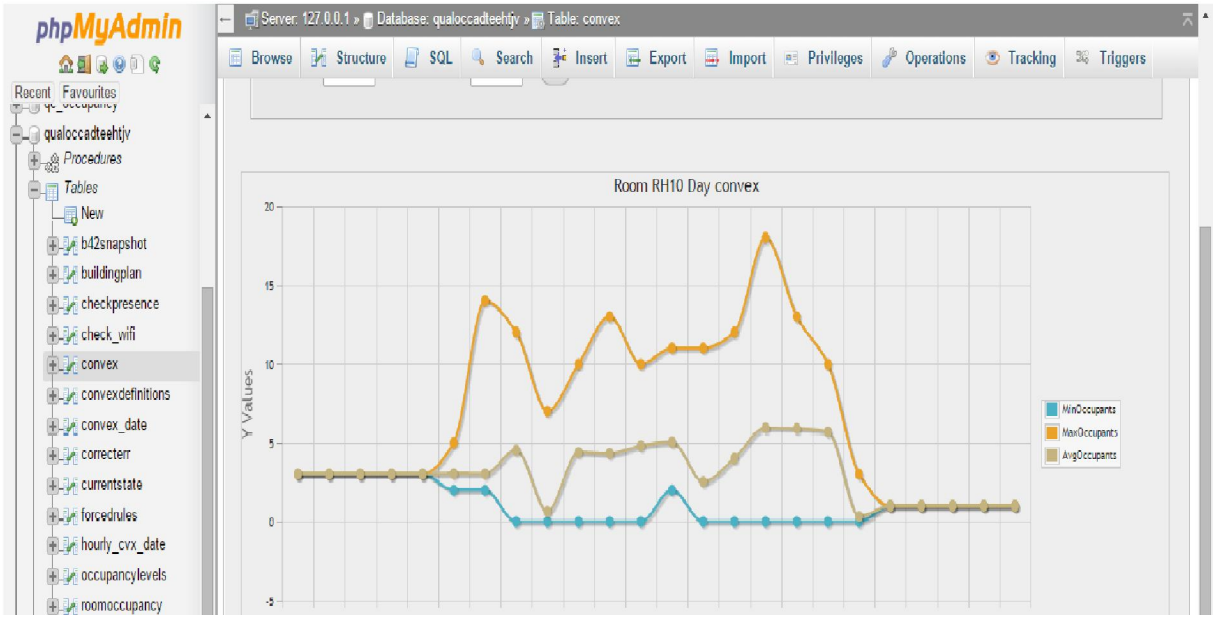
1. The Occupancy Engine **system receives data sent from different sensors** (door sensors, infrared room devices, Wi-Fi devices) using post method in http protocol. After format and logical validation all received data is stored in the database. Process diagrams are shown on Figure 4-6, Figure 4-7 and Fig 4-8.

2. **Quantify the number of people** in each room of a selected building, after each transition based on the information coming from different sensors.

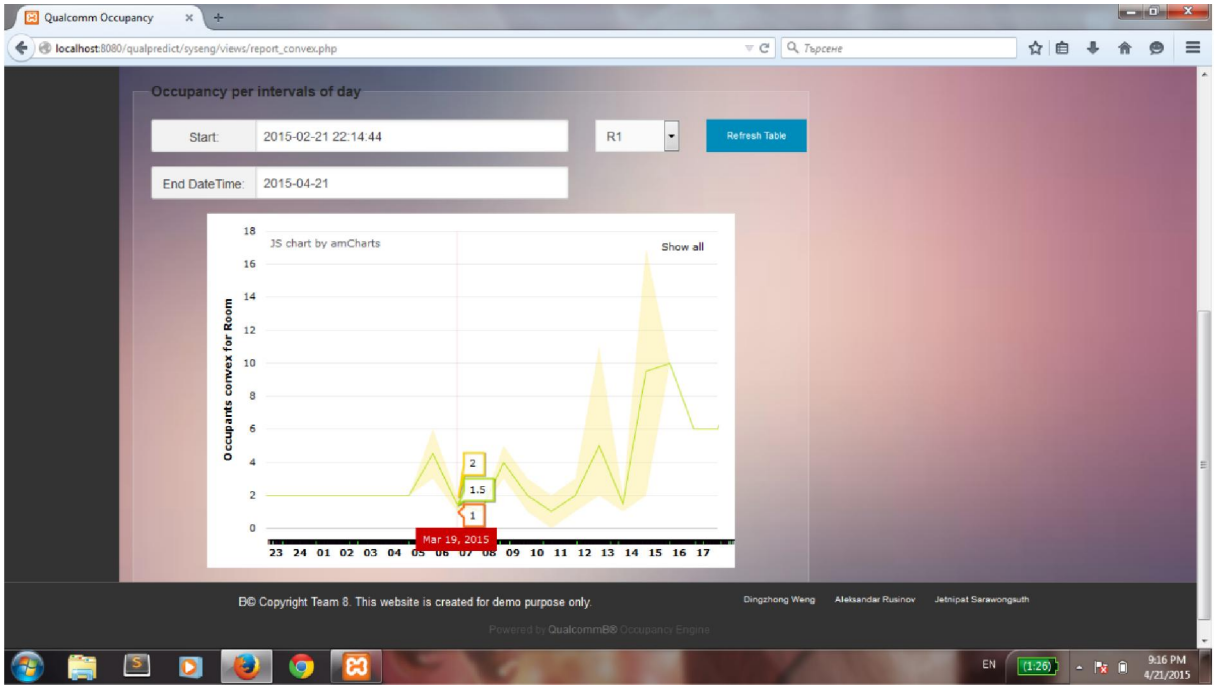
3. **Calculation of the confidence level** of the number of people in every room, based on the confidence, received from the door sensors. The idea is that the combined confidence level for a room is calculated by using all the confidence levels from each of the doors. We calculate this apriori confidence for the particular room using confidence for all doors which are connected to this room with the following formula: number of people who moved through particular door (in both directions) multiplied by the confidence for this door. This is summed for all doors that are connected to the particular room and finally divided by the number of all peoples moved in both directions through these doors. Thus we receive the confidence level for the given room. This is the apriori confidence for the room. When enough historical data is collected, the system can calculate the confidence level for a room using the historical data stored in the system and making the automatic corrections, made from the system for the doors, connected to the same room.
4. **Error Handling Module** - according to our client requirements, the system needs to be able to make automatic correction of the errors with very minimal human interactions.
   - The error checking and correction procedures are executed after every transmission of the data to ensure that any errors or inconsistencies are identified and taken care of right away. After checking the received data, the system automatically makes all necessary error corrections. For example, if the current-state table says that there is negative number of people in the room, the system will make an automatic correction and the negative value will not be stored in the current-state table. However, this does not mean that these inconsistencies and errors are not recorded. They will be recorded in as a historical data that we can use for further analysis. All corrections that are automatically corrected by the system will also be stored in the error corrections log, that way we should gather statistics for the errors and the system could use this data later to update the confidence level automatically if the errors reoccur.

- Another checking procedure is scheduled at pre-set time at night. On the base of information from door sensors and room devices. This procedure checks and compares occupancy in the rooms with the pre-defined set of rules. Using these rules, the system will be able to determine any unusual activities in a room and correct them, tracing back to the entrance/exit of the building using breath first search algorithm and tracking back the transitions from that room all the way to the entrance using calculating path. Later on, when there are enough data for analyses, the system can then create its own set of rules. The system will make an automatic correction by doing a breath first search and tracking back the transitions from that room all the way to the entrance. The process flow diagram is shown on Figure 4-10.
- Checking procedure for errors comparing the room sensor data and the door sensors data – making corrections where certain inconsistencies are found.
- User define set of rules (model) which program to follow when checking for errors, to determine whether or not an error needs to be corrected and how it should be corrected. We decided to create rules for the maximum number of people that could be in a certain room at a certain time of the day. These numbers are currently predefined for each room in each building. Later on, the system could create its own set of rules on the base of historical data.
- All automatic error corrections, made by the system, are stored in the correct error log table.





5. **The room occupancy pattern** – the occupancy convex procedure is used to find the pattern of the room occupancy - people presence in each room at certain time intervals. We calculate the minimum, maximum and the average (mean) number of people in each room at a certain period of time (parameter) based on data for 10 weeks(also parameter). Results are stored in a table and may be used for a smart error correction and reports base. Process of convex calculation is shown on Figure 4-9.



6. **Responsive design User interface with**:

6.1. Building plan displays the real time data from the database – current number of people in every room of the building

6.2. Reports for:

- Analyses based on the historical data - current and past occupancy along with presence confidence level, with possibility for the user to select room, door, past period time interval.
- Reports for occupancy (pattern) convex

Convex- occupancy pattern of presence for a past period

Occupancy pattern for day of week. Graphical report allows drill down for periods.

7. Java **simulator for sending online data from hardware sensors**. Door sensors simulator, generates random transitions through different doors in a building. The data, generated from the simulator, is sent to the database and stored in there just the same way as the real sensors would do.
8. PHP module to create and load test samples of historical data.

7. MANAGEMENT OF THE PROJECT INCLUDING WORK PACKAGES COMPLETED BETWEEN THE TEAM MEMBERS

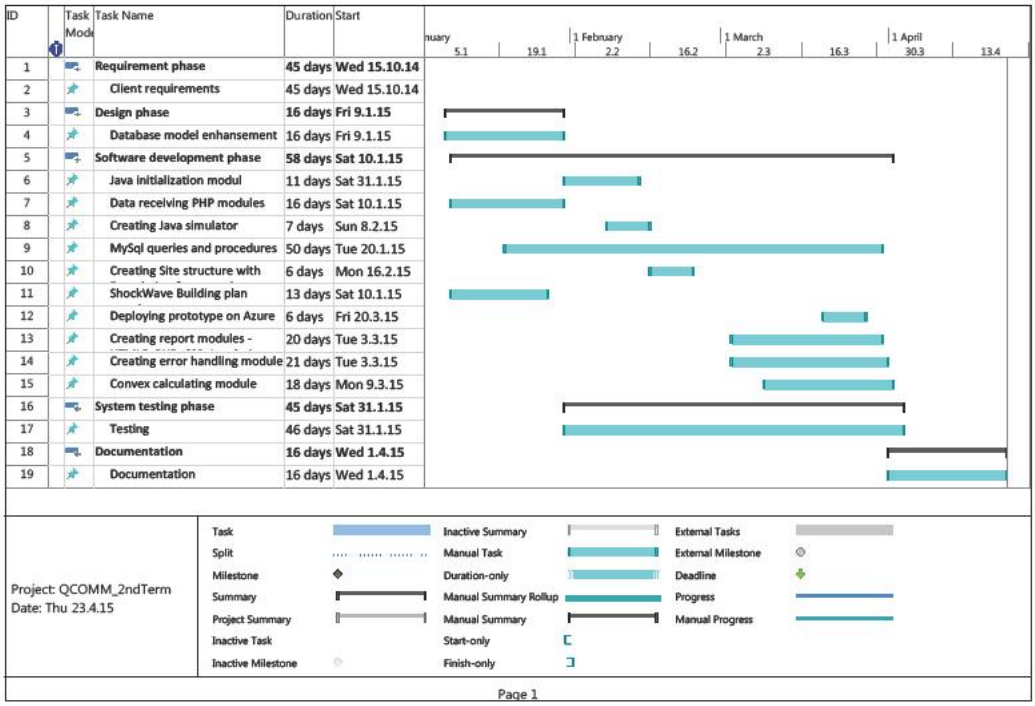
7.1. Requirements Management Plan

Weekly meetings with client to report the progress and to obtain feedback and comments.

7.2. Schedule Control Plan

Team leader has maintained a summary schedule showing the expected date of each milestone. After every team's meetings, the team leader re-evaluated the progress of the project to determine whether or not the project is on schedule. If the project was not on schedule, the team leader made consultation with team members to get the project back on track. This resulted in updating the schedule and/or re-evaluating the certain functionalities of the system.

7.3. Management plan



ID	Task Mode	Task Name	Duration	Start
1		**Requirement phase**	**45 days**	**Wed 15.10.14**
2		Client requirements	45 days	Wed 15.10.14
3		**Design phase**	**16 days**	**Fri 9.1.15**
4		Database model enhansement	16 days	Fri 9.1.15
5		**Software development phase**	**58 days**	**Sat 10.1.15**
6		Java initialization modul	11 days	Sat 31.1.15
7		Data receiving PHP modules	16 days	Sat 10.1.15
8		Creating Java simulator	7 days	Sun 8.2.15
9		MySql queries and procedures	50 days	Tue 20.1.15
10		Creating Site structure with	6 days	Mon 16.2.15
11		ShockWave Building plan	13 days	Sat 10.1.15
12		Deploying prototype on Azure	6 days	Fri 20.3.15
13		Creating report modules -	20 days	Tue 3.3.15
14		Creating error handling module	21 days	Tue 3.3.15
15		Convex calculating module	18 days	Mon 9.3.15
16		**System testing phase**	**45 days**	**Sat 31.1.15**
17		Testing	46 days	Sat 31.1.15
18		**Documentation**	**16 days**	**Wed 1.4.15**
19		Documentation	16 days	Wed 1.4.15

7.4. Work packages

N#	Tasks	Stage	Description	Assignee
Module 1 – Data initialization				
1.1	Initializing Java module - for set up, database creation, configuration and initial data loading in system parameter tables.	Finished and successfully tested. Loaded in GitHub repository.	Eclipse java project with clear packaged content involving Google Guava libraries, WindowBuilder, Java swing, Junit and MySQL connector	Aleksandar Rusinov
1.2	Java Client side module - Simulator of sending data online from hardware sensors	Finished and successfully tested. Loaded in GitHub repository.	Eclipse java project using JSON simple and Java net. OccupancyStatus.java-API class Simulator.java-simulator core Triple – utility class Junit test section	Aleksandar Rusinov
1.3	PHP module to create and load test samples of historical data, to populate tables with correct and consistent data.	Finished and successfully tested. Loaded in GitHub repository.	fill_csv_file.php t_createroomocc.php	Aleksandar Rusinov
	Module 2 – Data gathering and maintenance			
2.1	PHP modules to receive and process data from hardware sensors (door, room and WiFi sensors) using HTTP protocol and post/get methods. Format and logical validation of input data.	Finished and successfully tested. Loaded in GitHub repository.	add_data.php add_data_array.php add_data_rooms.php add_data_rooms_array.php add_data_wifi.php add_data_wifi_array.php	Aleksandar Rusinov
2.2	Procedure for checking and automatic error corrections, updating currentstate table, and in case of error insert correction record into error_correction log table	Finished and successfully tested.	Updatecurrentstate SQL stored procedure executed after every transmission of data	Aeksandar Rusinov
	Module 3 – Procedures and Functions for occupancy calculation			
3.1	Scheduled procedure to store room occupancy snapshots on a	Finished and successfully tested. Loaded in GitHub	store_currentstate.php	Aleksandar Rusinov

	predefined intervals	repository.		
3.2	Queries, views, procedures, functions calculating presence and movement for rooms and doors for current and past period	Finished and successfully tested. Loaded in GitHub repository	get_convex.php get_currentstate.php get_currentstate_inpast.php get_doors.php get_history_door.php get_history_door_usage.php get_history_room.php get_history_room_usage.php get_occ.php get_rooms.php	Aleksandar Rusinov
3.3	Calculate convex occupancy for pre-define past period and time intervals	Finished and successfully tested. Loaded in GitHub repository	fill_convex.php	Aleksandar Rusinov
Module 4 – UI and Reports module				
4.1	Building Plan with real time updates and Occupancy levels	Done, tested and implemented	Shockwave flash building plan done with Action Script that interacts with the server API and Shows the current state of the occupancy in the Building	Dingzhong Weng
4.2	Applying Responsive Design to the UI	Done and implemented	Using the new Fondations Framework in the web site layout	Dingzhong Weng
4.3	Usage of Ajax with JavaScript to fetch and display data	Done	report.js	Dingzhong Weng
4.4	Site structure	Done and implemented, tested menus and layout	**Dir structure** assets fonts forum include layout lib resources views	**Dingzhong Weng**
Module 5 – Error handling module				
5.1	Error handling module reports	Finished and successfully tested. Loaded in GitHub repository	errorhandling.php report_errors_alldoors.php report_errors_period.php get_history_errors_door.php get_history_error_door_usage.php	Aleksandar Rusinov
5.2	Scheduled procedure for automatic error checking and correction	Finished and successfully tested. Loaded in GitHub repository	checkrules.php drag_occupants.php	Aleksandar Rusinov
6 Testing				
6.1	Engine testing with	completed	Testings during lab sessions and	Jetnipat

	real data from Arduino sensors		during trials made by the hardware team	Sarawongsuth
6.2	Tests on the base of this emulation: tested error handling module, convex calculation procedure, queries and reports.	completed	Loaded data for past 10 weeks period, simulating real movement in an office building (about 15000 records in snapshot) Using PHP module for loading of historical data (fill_csv_file.php t_createroomocc.php)	Jetnipat Sarawongsuth
6.3	Tests with online simulator - building plan and reports	completed	Simulator of sending data online from hardware sensors	Jetnipat Sarawongsuth
6.4	Java Testing	completed	Implemented Junit tests in the Java modules	Aleksandar Rusinov
6.5	Tests following predefined use cases, sending data from csv files and using real http transfer mechanism	completed	Test_simulator.php Test_simulator_rooms.php Test_simulator_wifi.php	Jetnipat Sarawongsuth
Documentation				
7	Documentation	completed	Main Project Documentation, Bi-weekly reports, Test cases documentation, User Manual, System Manual	Jetnipat Sarawongsuth Aleksandar Rusinov

7.5. Configuration Management Plan

Configuration Management for software artifacts is handled using GitHub repository with version and source control to provide environment for team collaboration. From GitHub repository iterations of product is deployed on Azure and local host staging environment.

7.6. Testing and Quality Assurance Plan

All deliverables have gone through the appropriate review process. The review is required to ensure that each deliverable is of acceptable quality going through unit and integration tests and control. Appropriate test cases were elaborated and executed.

8. REFERENCED MATERIALS CITED AND EXAMPLES/TRIALS MADE

Referenced material	URL link
Occupancy Monitoring using Environmental & Context Sensors and a Hierarchical Analysis Framework Aftab Khan, James Nicholson, Sebastian Mellor, Daniel Jackson, Karim Ladha, Cassim Ladha, Jon Hand, Joseph Clarke, Patrick Olivier, Thomas Plotz Culture Lab, School of Computing Science, Newcastle University, Newcastle upon Tyne, UK. Energy Systems Research Unit, University Of Strathclyde, Glasgow, Scotland, UK.	http://di.ncl.ac.uk/publications/Khan-et-al-OccupancyMonitoringUsingEnvironmentalContext%20Sensors.pdf
A Sensor-Utility-Network Method for Estimation of Occupancy in Buildings Sean Meyn Electrical and Comp. Eng. and the Coordinated Sciences Laboratory University of Illinois at Urbana-Champaign Amit Surana, Yiqing Lin, Stella M. Oggianu, Satish Narayanan and Thomas A. Frewen United Technologies Research Center 411 Silver Lane, East Hartford, CT.	http://www.meyn.ece.ufl.edu/archive/spm_files/Papers_pdf/CDC09_2928_FI.pdf
A thesis submitted in partial fulfilment of the requirements of De Montfort University for the degree of Doctor of Philosophy Institute of Energy and Sustainable Development De Montfort University, Leicester, United Kingdom	www.dora.dmu.ac.uk/bitstream/handle/2086/10103/e-thesis%20Submission----Tobore%20Ekwevugbe----.pdf
Foundation zurb framework	http://foundation.zurb.com
Software development plan	http://acqnotes.com/acqnote/careerfields/software-development-plan
Project Management in a RUP Environment: Driving Iterative Development With Use Cases Ian Spence IBM Software Group | Rational software	http://www.bcs.org/upload/pdf/101203-projectmgt.pdf
Confidence interval	http://en.wikipedia.org/wiki/Confidence_interval
Linear regression	http://en.wikipedia.org/wiki/Linear_regression
linear least squares	http://en.wikipedia.org/wiki/Ordinary_least_squares
nonlinear regression	http://en.wikipedia.org/wiki/Nonlinear_regression
Curve Fitting with Linear and Nonlinear Regression Jim Frost	http://blog.minitab.com/blog/adventures-in-statistics/curve-fitting-with-linear-and-nonlinear-regression

9. TESTING AND EVALUATION OF THE PRODUCT SOLUTION

1. We successfully tested with the hardware devices.
2. We also made tests following predefined use cases, sending data from csv files and using real http transfer mechanism.
3. We made PHP module for loading of historical data for past 10 weeks period, simulating real movement in an office building (about 15000 records in snapshot) to test system performance. On the base of this emulation we tested error handling module, convex calculation procedure, queries and reports.
4. We also made online Java simulator to send data from different rooms of a building for testing, evaluation and demo purposes.

9.1. Tests from hardware sensors

Test 1.1. Test from door sensors

DoorId	event_time	transition	Confidence	Room Occupancy	Correct
D1	3/10/2015 14:09	1	97.00	1	YES
D1	3/10/2015 14:16	1	87.00	2	YES
D1	3/10/2015 14:42	1	92.00	3	YES
D1	3/10/2015 14:59	-1	98.00	2	YES
D1	3/10/2015 15:16	1	94.00	3	YES
D1	3/10/2015 15:20	1	86.00	4	YES
D1	3/10/2015 15:47	-1	89.00	3	YES
D1	3/10/2015 16:07	1	95.00	4	YES
D1	3/10/2015 16:18	-1	90.00	3	YES
D1	3/10/2015 16:29	1	92.00	4	YES
D1	3/10/2015 16:45	1	98.00	5	YES
D1	3/10/2015 16:55	-1	93.00	4	YES
D1	3/10/2015 17:09	-1	85.00	3	YES
D1	3/10/2015 17:15	1	94.00	4	YES
D1	3/10/2015 17:26	-1	96.00	3	YES
D1	3/10/2015 17:33	1	84.00	4	YES
D1	3/10/2015 17:37	1	96.00	5	YES
D1	3/10/2015 17:57	-1	92.00	4	YES
D1	3/10/2015 18:28	-1	93.00	3	YES
D1	3/10/2015 18:40	1	91.00	4	YES
D1	3/10/2015 18:45	-1	83.00	3	YES
D1	3/10/2015 18:51	-1	87.00	2	YES
D1	3/10/2015 18:56	-1	95.00	1	YES
D1	3/10/2015 19:17	1	92.00	2	YES
D1	3/10/2015 19:19	-1	95.00	1	YES
D1	3/10/2015 19:28	-1	96.00	0	YES
D1	3/10/2015 19:36	-1	60.00	-1	NO(stored in error correcton)
D1	3/10/2015 19:41	1	98.00	1	YES
D1	3/10/2015 19:53	-1	91.00	0	YES
D1	3/10/2015 19:57	1	83.00	1	YES
D1	3/10/2015 19:59	-1	94.00	0	YES
D1	3/10/2015 20:01	1	85.00	1	YES

Test 1.2 Test from room sensors

RoomID	DeviceID	HasPresence	Confidence	MeasuredAtTime
R1	W1	1	80.00	2015-03-10 14:36:36
R1	W1	1	80.00	2015-03-10 14:38:36
R1	W1	1	80.00	2015-03-10 14:40:36
R1	W1	1	80.00	2015-03-10 14:42:36
R1	W1	1	80.00	2015-03-10 14:44:36
R1	W1	1	80.00	2015-03-10 14:46:36
R1	W1	1	80.00	2015-03-10 14:48:36
R1	W1	0	80.00	2015-03-10 14:50:36
R1	W1	1	80.00	2015-03-10 14:52:36
R1	W1	1	80.00	2015-03-10 14:54:36
R1	W1	1	80.00	2015-03-10 14:56:37
R1	W1	0	80.00	2015-03-10 14:58:36
R1	W1	1	80.00	2015-03-10 15:00:36
R1	W1	1	80.00	2015-03-10 15:02:36
R1	W1	1	80.00	2015-03-10 15:04:36
R1	W1	0	80.00	2015-03-10 15:06:36
R1	W1	1	80.00	2015-03-10 15:08:36
R1	W1	0	80.00	2015-03-10 15:10:37
R1	W1	1	80.00	2015-03-10 15:12:36
R1	W1	0	80.00	2015-03-10 15:14:36
R1	W1	1	80.00	2015-03-10 15:16:36

9.2. Predefined test cases, sending data from csv files and using real http transfer mechanism.

Project name: Qualcomm Occupancy Engine					
Test Case ID: T1					
Test Case name: Receive Data and Current State Calculation					
Module name: add_data, add_data_array.php					
Test data: TD.1					
Pre-condition: number of people in the rooms: RH10 = 8; R1 = 2; R2 =2					
Brief Description: Test for the correct inserting in the database of a data, received from the door sensors. Test the correct calculation of current number of people in every room. Check the correct transfer of door movement into the movement for a room(in/out). Door 'D1' is between rooms 'RH10' and 'R1'; Door 'D10' is between rooms 'RH10' and 'R2' (see building plan description– App.6)					
St.	Action	Expected result	Actual result	Status	Comment
1.	Send data for movement of two people through door D1 with confidence 95,2%, now - 18-04-2015 10:25				

	Insert into snapshot table	D1, 2, 95,2%, 18-04-2015 10:25	D1, 2, 95,2%, 18-04-2015 10:25	Pass	Correct
	Update current state table	RH10 =6; R1 = 4	RH10 = 6; R1 = 4	Pass	Correct
	View for room movement	RH10, -2, 18-04-2015 10:25 R1, 2, 18-04-2015 10:25	RH10, -2, 18-04-2015 10:25 R1, 2, 18-04-2015 10:25		Correct
2.	Send data for movement of two people through door D10 in reverse direction (minus 2) with confidence 96,8%, now - 18-04-2015 10:25				
	Insert into snapshot table	D10, -2, 96,8%, 18-04-2015 10:25	D1, -2, 96,8%, 18-04-2015 10:25	Pass	Correct
	Update current state table	RH10 =8; R2 = 0	RH10 = 8; R2 = 0	Pass	Correct
	View for room movement	RH10, 2, 18-04-2015 10:28 R2, -2, 18-04-2015 10:28	RH10, 2, 18-04-2015 10:28 R2, -2, 18-04-2015 10:28	Pass	Correct

Project name: Qualcomm Occupancy Engine

Test Case ID: T1.1

Test Case name: Check sent data for format conformance and logical control

Module name: add_data, add_data_array.php

Test data: TD.1.1

Pre-condition: none

Brief Description: Test for the pattern and logical filtering of door sensors data. Send malformed data sets, bad formatted fields, logical incorrect data – i.e. datetime 32-04-2015 25:61:61, not existing in database doorId etc.

St.	Action	Expected result	Actual result	Status	Comment
	Send bad formatted/logical incorrect data to add_data.php				
1.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("transition is NOT numeric: DoorID:".$doorID."Time: ".date('Y-m-n H:i:s'))	error_log("transition is NOT numeric: DoorID:".$doorID."Time: ".date('Y-m-n H:i:s'))		
2.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid date DoorID:".$doorID."Time: ".date('Y-m-n H:i:s'));	error_log("Invalid date DoorID:".$doorID."Time: ".date('Y-m-n H:i:s'));		
3.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid	error_log("Invalid		

		confidence DoorID:".$doorID." confidence: ".$confidence);	confidence DoorID:".$doorID." confidence: ".$confidence);		
4.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid DoorID:".$doorID."Time: ".date('Y-m-n H:i:s'));	error_log("Invalid DoorID:".$doorID." Time: ".date('Y-m-n H:i:s'));		

Project name: Qualcomm Occupancy Engine					
Test Case ID: T2					
Test Case name: Receive data from the room devices					
Module name: add_data_rooms.php, add_data_array_rooms.php					
Test data: TD.2					
Pre-condition: none					
Brief Description: Test for the correct receive and store in the database the data from the room devices. RoomId, DeviceId, HasPresence, Confidence, MeasuredAtTime.					
St.	Action	Expected result	Actual result	Status	Comment
1.	Send data for a presence from the room R1, device P1, confidence 99%, now - 18-04-2015 10:40				
	Insert into Checkpresence table	D1,P1,1,99, 18-04-2015 10:40	D1,P1,1,99, 18-04-2015 10:40	Pass	Correct

Project name: Qualcomm Occupancy Engine					
Test Case ID: T2.1					
Test Case name: Check sent data from room sensors for format conformance and logical control					
Module name: add_data_rooms.php, add_data_array_rooms.php					
Test data: TD2.1					
Pre-condition: none					
Brief Description: Test for the pattern and logical filtering of room sensors data. Send malformed data sets, bad formatted fields, logical incorrect data – i.e. datetime 32-04-2015 25:61:61, not existing in database roomId etc.					
St.	Action	Expected result	Actual result	Status	Comment
	Send bad formatted/logical incorrect data to add_data_rooms.php				
1.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("transition is NOT numeric: ".$_SERVER['PHP_SELF']." RoomID:".$roomID." Time: ".date('Y-m-n H:i:s'))	error_log("transitio n is NOT numeric: ".$_SERVER['PHP_SELF']." RoomID:".$roomID. "Time: ".date('Y-m-n H:i:s'))		

2.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid date ".$_SERVER['PHP_SELF']." roomID:".$roomID." Time: ".date('Y-m-n H:i:s'));	error_log("Invalid date ".$_SERVER['PHP_SELF']." roomID:".$roomID. "Time: ".date('Y-m-n H:i:s'));		
3.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid confidence ".$_SERVER['PHP_SELF']." roomID:".$roomID." confidence: ".$confidence);	error_log("Invalid confidence ".$_SERVER['PHP_SELF']." roomID:".$roomID. "confidence: ".$confidence);		
4.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid roomID:".$roomID." Time: ".date('Y-m-n H:i:s'))	error_log("Invalid roomID:".$roomID. "Time: ".date('Y-m-n H:i:s'))		

Project name: Qualcomm Occupancy Engine					
Test Case ID: T3					
Test Case name: Receive and store data from the wi-fi devices					
Module name: add_data_wifi.php, add_data_array_wifi.php					
Test data: TD3					
Pre-condition: none					
Brief Description: Test for the correct receiving and inserting in the database of a data from the wi-fi devices. RoomID, DeviceID, MeasuredAtTime, NumberOfDevices.					
St.	Action	Expected result	Actual result	Status	Comment
1.	Send data for two wi-fi devices from the room R1, device W1, now - 18-04-2015 10:50				
	Insert into check wifi table	R1,W1,18-04-2015 10:50, 2	R1,W1,18-04-2015 10:50, 2	Pass	Correct

Project name: Qualcomm Occupancy Engine
Test Case ID: T3.1
Test Case name: Check sent data from wi-fi sensors for format conformance and logical control
Module name: add_data_wifi.php, add_data_array_wifi.php

Test data: TD3.1					
Pre-condition: none					
Brief Description: Test for the pattern and logical filtering of room sensors data. Send malformed data sets, bad formatted fields, logical incorrect data – i.e. datetime 32-04-2015 25:61:61, not existing in database roomId etc.					
St.	Action	Expected result	Actual result	Status	Comment
	Send bad formatted/logical incorrect data to add_data_rooms.php				
1.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("numberOfDevices is NOT numeric: ".$_SERVER['PHP_SELF']." RoomID:".$roomID." numberOfDevices: ".$numberOfDevices);	error_log("numberOfDevices is NOT numeric: ".$_SERVER['PHP_SELF']." RoomID:".$roomID."numberOfDevices: ".$numberOfDevices);		
2.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid date ".$_SERVER['PHP_SELF']." roomID:".$roomID." measuredAtTime: ".$measuredAtTime)	error_log("Invalid date ".$_SERVER['PHP_SELF']." roomID:".$roomID." measuredAtTime: ".$measuredAtTime)		
3.	Insert into snapshot table	No action	No action	Pass	Correct
	Php_error_log file	error_log("Invalid roomID:".$roomID."Time: ".date('Y-m-n H:i:s'));	error_log("Invalid roomID:".$roomID."Time: ".date('Y-m-n H:i:s'))		

Project name: Qualcomm Occupancy Engine
Test Case ID: T4
Test Case name: Check-errors and correct-errors procedure
Module name: check_error
Test data: TD4
Pre-condition: number of people in the rooms: R2 = 2; R6 = 5
Brief Description: Test the check-error and correct- error procedure when receiving data from the door sensors. The system performs automatic check and correct errors procedures using specific algorithms. Automatic procedures for checking errors and making errors correction are executed after every transmission of data. In this test we simulate sending of wrong data from the door sensors, trying to extract more people than the number of people currently in the room, and that way to receive negative number of people in the current state of the room. The system should make an automatic correction and the negative value shouldn't be stored in the current-state table. All transitions, coming from the door sensors must be stored in the snapshot

table (history log). All corrections, automatically done by the system, also must be stored in the error corrections log. Test the correct calculation of current number of people in every room after transition. Check the correct transfer of door movement into the movement for a room(in/out). Door 'D8' is between rooms 'R2' and 'R6' (see building plan description– appl.6)
('D8', 'R2', 'R6'),

St.	Action	Expected result	Actual result	Status	Comment
1.	Send data for a movement of three people through door D8 with confidence 95,2%, now - 18-04-2015 14:12				
	Insert into snapshot table	D1, 3, 95,2%, 18-04-2015 14:12	D1, 3 , 95,2%, 18-04-2015 14:12	Pass	Correct
	Insert into correct error log table	D1, -1, 95,2%, 18-04-2015 14:12	D1, -1, 95,2%, 18-04-2015 14:12	Pass	Correct
	Update current state table	R2 =0; R6 = 7	R2 =0; R6 = 7	Pass	Correct
	View for room movement	R2, -2, 18-04-2015 14:12 R6, 2, 18-04-2015 14:12	R2, -2, 18-04-2015 14:12 R6, 2, 18-04-2015 14:12	Pass	Correct

Project name: Qualcomm Occupancy Engine

Test Case ID: T5

Test Case name: Schedule procedure for night error correction

Module name: check_rules

Test data: TD5

Pre-condition:

1. In the forced rules table - Forced rule that on Saturday for the time interval from 00:00 to 01:00, there should be minimum 0 and maximum 0 number of people in room R12, and this rule is 'forced'. (18-04-2015 is Saturday)
2. Current number of people in the room R12 = 2.

Brief Description: Test checking rules procedure, which is scheduled periodically at pre-defined time interval (i.e. every night at 00.00) and performs automatic checking and error corrections according to user-defined rules. User-defined rules are stored into the forced rules table – rules for min/max occupancy per interval of weekday. Using these rules, the system will be able to determine wrong activities in a room and correct them, tracing back to the entrance/exit of the building using breath first search algorithm. All automatic corrections, made by the system, must be stored in the error corrections log table in the database for further analyses. All data after these error corrections(back tracing) should remain consistent. System should make a correct calculation of the current number of people in every room after transition. Checking of the correct transfer of door movement into the movement for a room(in/out).
('D18', 'R12', 'R10'), ('D14', 'RH10', 'R10'), ('D0', '0000000000', 'RH10') (see the building plan App1)

D0-> door from outside to the corridor 'RH10'
D14- > door between the corridor 'RH10' and the room 'R10'
D18-> door between the room 'R12' and the corridor 'R10'

St.	Action	Expected result	Actual result	Status	Comment
1.	Scheduled procedure for				

	night error correction. scheduled periodically at pre-defined time interval (i.e. every night at 00.00) performs automatic checking and error corrections according to user-defined rules				
	Insert into correct error log table	D18, 2, 95,2%, 18-04-2015 00:00:00 D14, -2, 95,2%, 18-04-2015 00:00:05 D10, -2, 95,2%, 18-04-2015 00:00:10	D18, 2, 95,2%, 18-04-2015 00:00:00 D14, -2, 95,2%, 18-04-2015 00:00:05 D10, -2, 95,2%, 18-04-2015 00:00:10	Pass	Correct
	Update current state table	R12,0	R12,0	Pass	Correct

Test Data

TD1. //Door sensors

DoorID, transition, Confidence, event_time

D0-> door from outside to the corridor 'RH10'

D1- > door between the corridor 'RH10' and the room 'R1'

D10-> door between the corridor 'RH10' and the room 'R2'

Test data set:

Step 1: D1, 2, 95.20, NOW()

Step 2: D10, -2, 96,8%, NOW()

TD1.1 //Door sensors

DoorID, transition, Confidence, event_time

Test data set:

Step 1: D1, aa, 95.20, NOW()

Step 2: D10, -2, 101,8%, NOW()

Step 3: D10, -2, 101,8%, 2015-04-32 10:21:22

Step 4: R10, -2, 101,8%, NOW()

TD2. // RoomDevices

RoomId, DeviceId, HasPresence, Confidence, MeasuredAtTime

Test data set:

R1,P1,1,99,NOW

TD2.1 //Room sensors

RoomId, DeviceId, HasPresence, Confidence, MeasuredAtTime

Test data set:

Step 1: R1, P1, a1, 99, NOW

Step 2: R1,P1,1,99, 2015-04-32 10:21:22

Step 3: R1,P1,1,-99,NOW

Step 4: D1,P1,1,99,NOW

TD3. //WiFi

RoomID, DeviceID, MeasuredAtTime, NumberOfDevices

Test data set:

R1,W1,NOW,2

TD3.1 //WiFi

RoomID, DeviceID, MeasuredAtTime, NumberOfDevices

Test data set:

Step 1: R1,W1,NOW,dd

Step 2: R1,W1, 2015-04-32 10:21:22,1

Step 3: Z1,W1,NOW,1

TD4. // Door sensors - send data for a movement of three people through door D8 with confidence 95,2%, now – (18-04-2015 14:12) (building plan: 'D8', 'R2', 'R6'),

DoorID, transition, Confidence, event_time

Test data set:

D8,3,95.20,NOW

TD5. // Schedule procedure for night error correction

forcedrules table (user-defined rules for min/max occupancy per interval of weekday to be used in scheduled procedure for night error correction). Test data means that on Saturday for the time interval from 00:00 to 01:00, there should be minimum 0 and maximum 0 number of people in room R12, and this rule is 'forced'.

RuleID, RoomID, DayOfWeek, FromTime, ToTime, MaxOcupancy, MinOcupancy, MustBeForced

Test Data set:

104,R12, 6, 00:00,01:00,0,0,1

('D18', 'R12', 'R10'), ('D14', 'RH10', 'R10'), ('D0', '0000000000', 'RH10') (see the building plan TD6)

D0-> door from outside to the corridor 'RH10'

D14- > door between the corridor 'RH10' and the room 'R10'

D18-> door between the room 'R12' and the room 'R10'

Current state – current number of people in rooms. Test data means, that there are 2 people in room R12.

RoomID, NumberofPeople

Test Data set:

R12, 2

TD6.// Building plan configuration – doors and rooms

`buildingplan` (`DoorID`, `Room1ID`, `Room2ID`)

('D0', '0000000000', 'RH10'),

('D1', 'RH10', 'R1'),

('D10', 'RH10', 'R2'),

('D11', 'RH10', 'T14'),

('D12', 'R1', 'T14'),

('D13', 'RH10', 'R8'),

('D14', 'RH10', 'R10'),

('D15', 'RH10', 'R11'),

('D16', 'RH10', 'R12'),

('D17', 'R11', 'R10'),

('D18', 'R12', 'R10'),

('D19', 'RH10', 'T13'),

('D2', 'RH10', 'R3'),

('D20', 'RH10', 'T15'),

('D3', 'RH10', 'R4'),

('D4', 'RH10', 'R7'),

('D5', 'RH10', 'R5'),

('D6', 'RH10', 'R6'),

('D7', 'R2', 'R5'),

('D8', 'R2', 'R6'),

('D9', 'R5', 'R6');

10. SDK/COLLABORATION OPPORTUNITIES FOR FURTHER DEVELOPMENT

The first step is to build an inference machine for machine learning and occupancy prediction (i.e. based on neural network) consuming information from sensors, historical data for the building, correct errors history, a penalty function – based on initial predefined rules and extrapolated from room and Wi-Fi sensors along with error correcting data. To select a suitable approach, historic data might be analysed by "of-the-shelf" workbench for data analysis, data mining, computer learning and predictive modelling – for instance WEKA, ELKI, RapidMiner - just to mention some of available free open source platforms. The second step is to ensure that we have reliable results in an appropriate response time in case of big buildings and huge amount of historical data - by applying scalable decentralized algorithms and/or parallel computing. Here choice of model and platform depends of volume and dynamic of data – Apache Hadoop open-source software family of products look like good first choice for establishing reliable, scalable, distributed computing environment to start with.

11. SYSTEM MANUAL AND USER MANUAL

12. COMMUNITY FORUM