Using BLE Beacons For Determining Location On A Construction Site

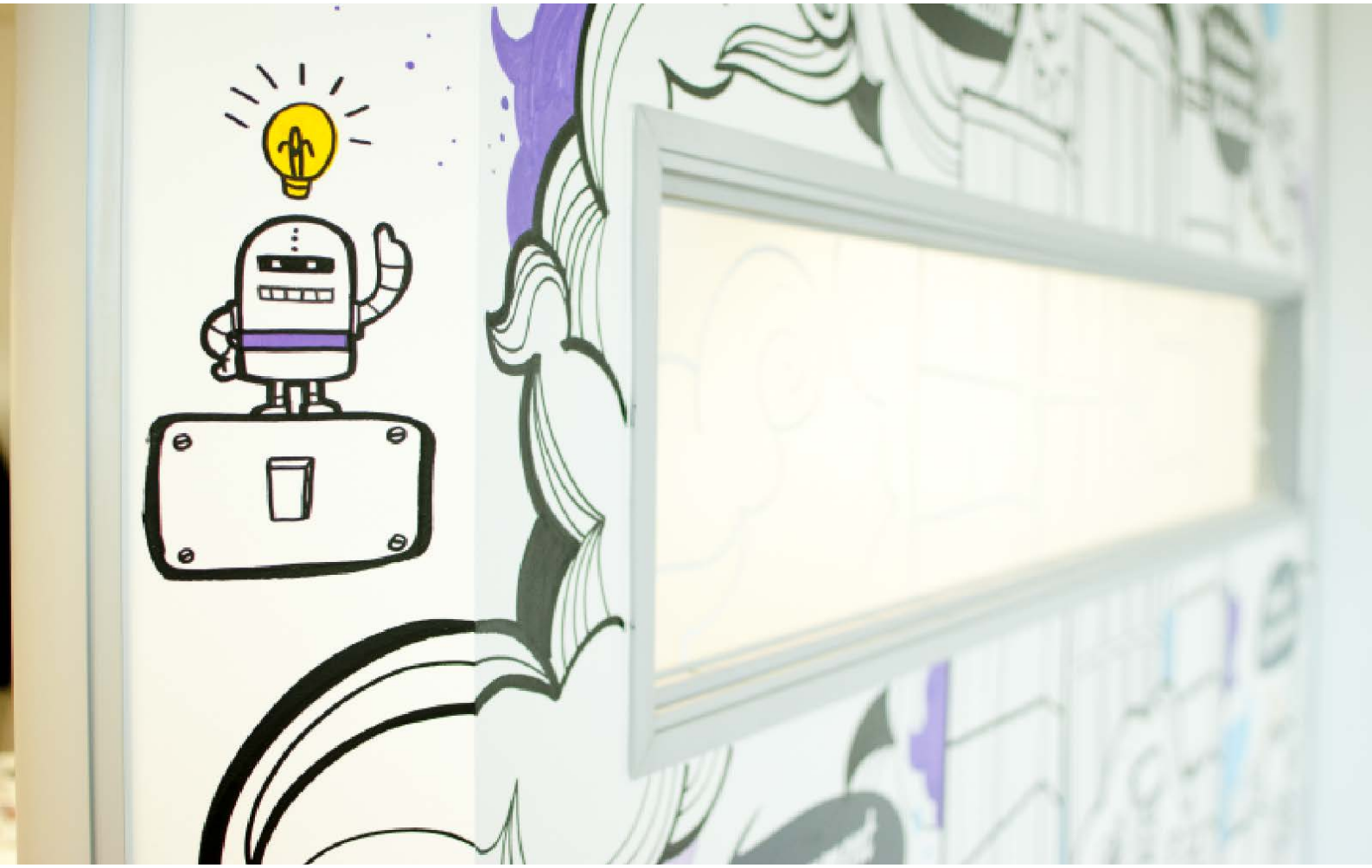
Crossrail

Version	Produced by	Date
2.0	Luke Stringer	August 14, 2015

3SQUARED.COM
+44 (0)333 121 3333
info@3squared.com
+44 (0)333 121 3330
@3squared

3SQUARED Ltd
Fountain Precinct, Balm Green, Sheffield S1 2JA
United Kingdom

USING **KNOWLEDGE**, **EXPERIENCE** AND **INNOVATION** TO DELIVER RESULTS.



Executive Summary

The initial brief of this project was to identify possible applications of Bluetooth Low Energy (BLE) beacons for general safety compliance in a construction site environment. By identifying use cases our objective was to develop software solutions to satisfy the core requirements.

The core requirements were to, as accurately as possible, pinpoint the whereabouts of a user on a floorplan and provide context aware information such as documentation, drawings and important information relevant to the user, based on their location.

Our research led to 3 possible approaches:

- Naïve Zone Detection
- Trilateration
- Fingerprinting

Each approach was evaluated and trilateration chosen as the most suitable for our purposes.

Test results are detailed in this report but in summary the test scripts were successful in determining a user's presence within a zone. The business logic applied to meet the core requirement of raising a notification if a user entered a restricted zone worked as expected and exceptions were raised at the web portal detailing instances when a zone had been breached.

There were areas where accuracy would have to improve before it could be used in a real world setting. For example, it was noted that when first positioned in a zone beacons took some time to stabilize. There were also instances where if a beacon was not detected the app seemed to hold onto the signal of the last detected beacon, which to the user would appear like the app had frozen. In fact the app was working as expected, it just wasn't 'confident' with the signal from the next beacon.

For the app to be used commercially as a safety compliance tool, further development would be recommended to improve accuracy:

- Work to eradicate occasional false positives that can arise when the app is searching for beacons.
- Filtering out weak beacon signals to determine which are nearest.
- Improve the UI/UX of the web application to make managing and editing zones simpler.
- Test the second generation of beacons now entering the market.

Introduction

Crossrail's Innovate18 programme provided a brief for a pilot project to trial Bluetooth Low Energy (BLE)technology to help understand the technologies capability and assess its feasibility in providing micro-location positional information and solving the challenge of accurate real time positioning on a construction site.

A proof of concept pilot project was considered an ideal opportunity to explore the capabilities of BLE beacons helping Crossrail to understand their limitations and feasibly assess the suitability of the technology for deployment in a live construction site.

It is thought that the learning and output from this pilot project could be used as an excellent risk mitigation tool for any organisation planning to implement micro location technology.

As part of the 3Squared project process a requirements gathering session was held where user stories were collected and prioritised using the MOSCOW method of prioritisation. Each requirement is categorised as a Must Have, a Should Have, a Could Have or a Won't Have.

The core requirements (must haves) were to create and manage zones within a space and enforce restrictions on zones based on a user's access levels. In practice this would mean a user would be excluded from entering a zone if their user was not permitted there. If they entered the restricted zone anyway an exception would be raised on the web portal, notifying relevant parties.

The other must have requirement from user stories was to restrict device hardware and software feature in a contextually aware way based on the zone or location. This requirement was quickly found to be impossible to implement due to limitations of the MDM solution. The two obstacles we found were:

1. MDM solutions don't provide APIs so it is not possible to send down configuration settings to a device based on what zone they are in.
2. A device must have a network connection to communicate with the MDM and this connectivity cannot be guarantee on a construction site.

The first field tests conducted at the Crossrail offices in December 2014 produced disappointing results due to signal instability and generally quite poor levels of accuracy but as part of our testing our Mobile Developer had researched several methods of collecting data from the BLE beacons and it was agreed we would implement a trilateration algorithm into the existing mobile app which it was hoped would improve accuracy.

For our second test at the Crossrail office the new algorithm was used and results were a marked improvement and it was decided by the project team to proceed with the next stage of testing. A test plan was created by 3Squared and it would be baselined in the Crossrail office first then the same test script run in construction site conditions at an agreed area within the Bond Street station site to compare results and draw any conclusions.

Requirements

A number of user stories were collected defining the requirements for the software. These were prioritised using the MoSCoW system and were designated as a “Must” / “Should” / “Could” / “Won't” feature to implement.

ID	As a	User Role	I want to	Requirement	So that I can	Outcome/Benefits	Priority
1	As a	H & S Manager	I want to	Set up zones and restricted access levels	so that I can	inform site ops of potential hazards	Must
2	As a	H & S Manager	I want to	Update and manage zones including levels of restrictions and be able to notify field operatives in a timely manor of these changes	so that I can	react quickly to changes in the construction site environment and notify field operatives of those changes.	Must
3	As a	Management	I want to	Restrict device hardware and/or software features and functions by zone or location	so that I can	manage safe use of devices on site	Must
4	As a	Site Based Personnel	I want to	Be able to safely use my mobile device in the field by having my device restricted to certain apps and features in a given context	so that I can	comply to regulations and be safe	Must
5	As a	H & S Manager	I want to	Manage and monitor controlled access to zones tied to job roles and competencies	so that I can	can define competency requirements for zones and help to control and manage access to zones on a site	Must
6	As a	Site Based Personnel	I want to	Receive real time safety alerts that are contextually aware i.e. time and zone	so that I can	avoid hazards and comply with safety regulations and be aware of evacuation procedures	Should
7	As a	Site Based Personnel	I want to	Be notified as I enter a restricted area	so that I can	avoid hazards and have the correct equipment, and also to be notified if I am competent to be in this area	Should
8	As a	H & S Manager	I want to	Produce exception reports	so that I can	proactively manage health and safety and instances of non conformance	Should
9	As a	H & S Manager	I want to	Tag H&S alerts to certain zones eg new activity starting and specify additional attributes such as start times and durations of activities.	so that I can	inform site ops of potential hazards	Should
10	As a	H & S Manager	I want to	Link zones to sound monitoring devices	so that I can	manage mandatory hearing protection zones and inform field ops of high decibel levels (ear protection)	Could
11	As a	H & S Manager	I want to	Analyse field operatives paths	so that I can	optimize construction zones	Could
12	As a	H & S Manager	I want to	Overlay data points onto a floor plan to create heatmaps	so that I can	analyse trends and generate insight	Could
13	As a	H & S Manager	I want to	On a floor plan, define emergency exits with routes	so that I can	enable field operatives to quickly and effectively find emergency exits based on their micro location	Could
14	As a	H & S Manager	I want to	Track the location of large assets such as pre-fab elements or plant.	so that I can	monitor the safe usage of equipment.	Could
15	As a	H & S Manager	I want to	Have access to emergency exits and routes near me	so that I can	can check whether they are clear and safe to use	Could
16	As a	Site Based Personnel	I want to	Receive information relevant to the given context	so that I can	make my job easier and make me more efficient	Could
17	As a	Site Based Personnel	I want to	Fill out forms with minimal input by having forms prepopulated with relevant information based on the given context	so that I can	Be more efficient and save time	Could
18	As a	Site Based Personnel	I want to	Be able to be alerted to dynamic danger zones eg. Slews and plant as I approach the danger zone	so that I can	stay safe	Could
19	As a	Site Supervisor	I want to	Remotely update zones from my device	so that I can	respond in real time to changes in the environment	Could
20	As a	Machine Operative	I want to	Be alerted when someone is in an exclusion zone or when someone has entered my zone	so that I can	be aware of my surroundings and keep others safe	Could
21	As a	Crane Operator	I want to	Be aware of when my slew is relative to a construction site and people on the site	so that I can	keep others safe in the field	Could
22	As a	Field Engineer	I want to	Receive documents relative to a given context eg my role, location, time etc.	so that I can	perform my role more efficiently	Could
23	As a	Field Engineer	I want to	Auto-populate and limit data entry forms with relevant information	so that I can	safe time inputting data	Could
24	As a	Field Engineer	I want to	Add location to photos	so that I can	provide more context to photos	Could
23	As a	Quality Manager	I want to	Engineers to fill out forms at the point of activity rather than elsewhere	so that I can	Be confident that data is captured at the point of event and not after	Could

Market Assessment

There are a number of software and hardware solutions that have been developed bearing relevance to this project. The use case for BLE beacons that Apple initially pushed was in retail environments. One example of this would be in a shopping mall where a mobile application could present location specific offers and information to a shopper depending on where they where in the mall. Other developments in this space include solutions providing turn-by-turn navigation to customers, which is something that is of particular relevance to this project.

Other areas of interest for this project are in the development of safety critical applications that use wireless technologies to track user's positions. A number of these applications have been developed by universities and their research papers provide details on the methodology behind their approaches.

Retail Applications

Aruba Networks and their partner Meridian have developed a hardware and software solution for indoor positioning using BLE beacons and Wi-Fi nodes. Their solution has been used at the Levi's Stadium in California to build a mobile application. This app allows users to buy and use parking passes, order food and drink and, importantly for us, provides "wayfinding to navigate around the building".

In this solution, Aruba battery powered beacons are first deployed throughout the Levi's Stadium - 1,100 beacons to cover the whole floor space. Then a lesser number of Aruba Wi-Fi nodes are also deployed. Afterwards a set of USB powered beacons are connected to the Wi-Fi nodes to allow the battery powered beacons to communicate directly with the Wi-Fi nodes. Then the Wi-Fi nodes send beacon data to a software controller system for battery monitoring, firmware updates etc.

Once the hardware is installed, Meridian's AppMaker Editor software is used to define a map and place points of interest at locations such as where food can be purchased. Additionally, zones can be specified so that a user can opt in to location specific notifications while they are exploring the stadium. Once this information has been defined the software is then able to provide turn-by-turn navigation instructions on a mobile device from any given point in the stadium to another.

This hardware and software solution looks promising in terms of what they have achieved at the Levi's Stadium. Their requirements seem quite similar to this project's however the main difference is that the product is used in a purely commercial environment without any concerns regarding safety.

Safety Critical Applications

Sunkyu Woo et al investigated the feasibility of a Wi-Fi based indoor positioning system for construction sites. A series of experiments were carried out in a shield construction site in China where a system using a technique known as "fingerprinting" was used to track the approximate location of labour. The requirements outlined in this paper align closely to the those of the Crossrail project both in terms of the deliverables and also the environment in which the product will be deployed.

The system developed here was based around the concept of building a digital fingerprint of the construction site before tracking could begin. Building the fingerprint involved collecting, storing, and interpolating RSSI readings from Wi-FI access points at numerous positions within the construction site. Once built this fingerprint was used as a way to

determine where a worker was by comparing the live RSSI readings being received with the ones stored in the fingerprint.

A key difference here is the choice of technology: the Crossrail project is to use BLE beacons whereas the experiments conducted in China used Wi-Fi access points. This is an important distinction to make as BLE beacons are not designed to be used for accurate positioning. In contrast the Received Signal Strength Indication (RSSI) from Wi-Fi access points give more reliable measurements of distance. Despite these caveats regarding the technology the research conducted by Sunkyu Woo et al is still of relevance due the similarities in project scope and requirements, and also the impressive results they managed to obtain: accuracy within 5m of error for the test site.

Methodology

Two key areas were investigated:

1. Device management to restrict services and features on a device based on when and where the user is.
2. Location computation using the BLE beacons.

Mobile Device Management

To satisfy requirements 3 and 4 it is necessary to restrict what a user can do on a device via software. This can be achieved by using an MDM (Mobile Device Management) solution. By installing MDM software on an iOS device, apps can be distributed wirelessly outside of the AppStore and restrictions defined to determine what a user can and cannot do with their device. These restrictions include disallowing access to certain websites, what files can be shared and how, and what apps can be accessed.

Configuration settings are defined in the MDM web application and then transferred to all managed devices via a configuration profile. Configuration settings can be time sensitive so they only apply during certain periods. To satisfy our requirements however it will be necessary to also make them location sensitive so that when a user travels into a specific zone a configuration profile is applied. In order to do this the MDM solution will need to provide a way for our software to programmatically generate and then communicate a profile to the device. For example, when a user walks into a zone where dangerous work is happening the mobile device will need to communicate this to the MDM (via another web application perhaps) and then the MDM will send down a profile to the device applying the necessary restrictions.

Through investigating various MDM solutions, it is unclear whether our requirement of programmatically applying a profile is achievable. Absolute MDM detail the ability to “silently install/remove apps using management APIs” as one of their capabilities, but we have been unable to find any specifics of what this will allow us to implement. Furthermore, even if it is possible to programmatically apply profiles, disabling hardware features such Wi-Fi connectivity is simple not possible. This is a limitation of iOS not of any particular MDM solution.

Another limiting factor in implementing device management is that it will be necessary for devices to be “online” at all times in order to receive the configuration profiles. A stable and reliable network connection has to be guaranteed and this is something that will not be possible on a construction site. Moreover, one of the reasons BLE beacons were chosen as a technology for investigation in the first place was that Wi-Fi connectivity on a construction site could not be relied upon. A possible workaround for this would be to build in all of the configuration profiles into the iOS application itself so that obtaining them over a network would not be necessary. However, this is not a workable solution as a standalone application cannot apply profiles directly to iOS - it has to be done via an MDM solution. Again, this is something that has been defined by Apple for legitimate security reasons and not a limitation of any particular MDM solution.

In conclusion it was decided that device management is a requirement that we unfortunately cannot fulfil.

Location Computation

For the iPad app to compute where it was located three approaches were identified: naïve zone detection, trilateration, and fingerprinting.

Naïve Zone Detection

The focus of the approach is to detect when the device running the iPad app is located inside a zone, rather than calculate its actual location / coordinate within the usage space. Here beacons were associated with a zone and the

detection of that zone by the app was defined as “being able to observe enough of the beacons”. More formally a zone would be detected if:

- n% of its beacons could be observed.
- Those beacons had a “close enough” proximity.

When observing beacons in an iOS app the Core Location API identifies beacons as having a “proximity”. From the Apple CLBeacon API documentation:

“The value in this property [proximity] gives a general sense of the relative distance to the beacon. Use it to quickly identify beacons that are nearer to the user rather than farther away.”

A proximity can have four different values:

- *Immediate*. Within a few centimetres of the device. The API is highly confident of this distance.
- *Near*. Within a few meters of the device. The API is fairly certain of this distance.
- *Far*. More than a few meters from the device. The API is not confident of this distance as the signal it too weak.
- *Unknown.* The API is able to provide any usable data for the beacon.

An implementation of this approach would be to choose a value for the n% and define how close those beacons would need to be. For example, if 66% of a zone’s beacons were observed to have a *Near* or closer proximity then it would calculated that the device were inside the zone.

The value of n% and the necessary proximity could be tweaked through trial and error to discover the optimal values to maximize accuracy.

Trilateration

If the distance from 3 beacons at 3 known locations in a 2D coordinate space can be obtained then the device’s location can be calculated using a mathematical technique called Trilateration. This works by plotting the 3 distances as circles, where the radius of each circle is equal to the distance from each beacon, and then finding the intersection. This intersection is the position of the device - the 2D coordinate that falls on the circumference of each of the circles.

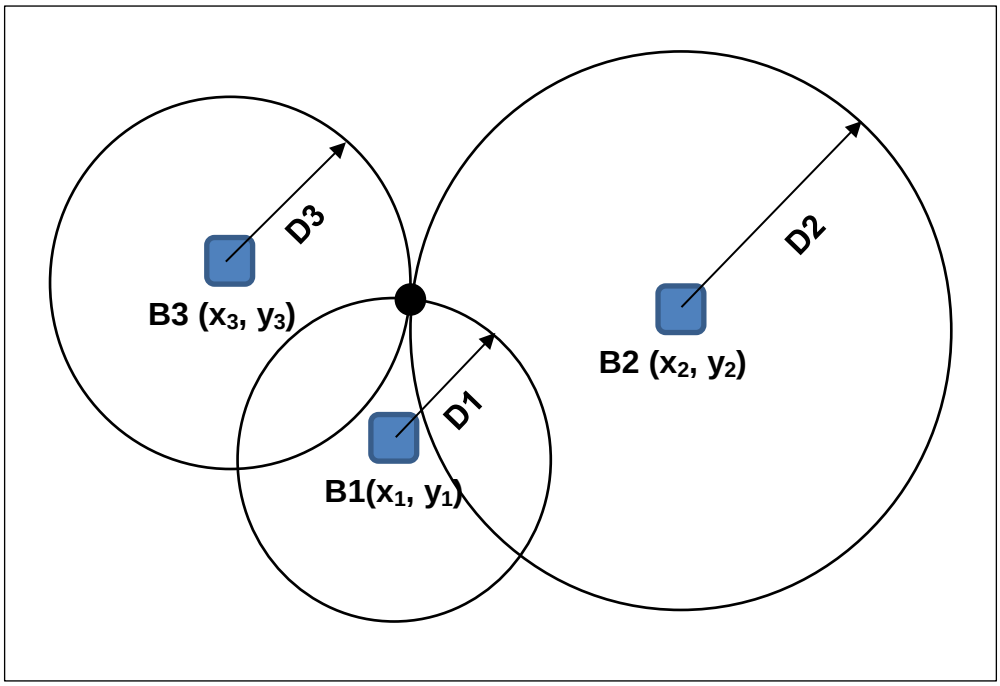


Diagram 1: Using Trilateration for position calculation.

Diagram 1 demonstrates how trilateration works. Given we have 3 beacons B1, B2, B3, at known positions (x_1, y_1), (x_2, y_2), (x_3, y_3), which are observed to be distances D1, D2, D3 away from a position, we can calculate the coordinate of that position as the intersection of the 3 circles.

The CLBeacon API provides an estimate of the distance from a beacon. To use the API terminology this value is known as the "accuracy" and is defined as follows:

"[the accuracy value] indicates the one sigma horizontal accuracy in meters. Use the property to differentiate between beacons with the same proximity value. Do no use it to identify the precise location for the beacon. Accuracy values may fluctuate due to RF interference."

Apple clearly indicates that the value they provide for distance should not be used to perform precise calculations. This is due to radio signals having a wide degree of variability when reflected by obstacles and refracted around corners, which is especially prevalent indoors. The human body, being mainly water, also hugely affects signals. Whether the device is in a pocket or bag, or somebody is blocking the path from a device to beacon all affect the precision of the distance value.

Using imprecise distances with trilateration will not yield a single point of intersection. Therefore the calculated position of the device will also be imprecise.

To minimise this problem we can ensure the app running the trilateration algorithm is in foreground of iOS (not in the background) and the device is not in a pocket or bag. This will enable the app to get the beacon distance readings as frequently as iOS can provide - every second at maximum. We can also simply use more beacons – this will increase the likelihood of the device being nearer to a beacon and getting a stronger signal allowing iOS to provide a more precise distance reading for our trilateration calculation.

Fingerprinting

This third technique is outlined in detail by Woo et all in “Application of WiFi-based indoor positioning system for labour tracking at construction sites: A case study in Guangzhou MTR”.

The fingerprinting approach can be broken down into 2 phases:

1. **Training phase** to build a digital “fingerprint” of a space.
2. **Tracking phase** to process signal data in real time.
   - Compare live data with the store RSSI sample points.
   - Determine position by finding the optimal match between live and stored data.

During the training phase a large set of RSSI (received signal strength indicator) readings are collected from beacons as predefined and known locations. These readings are then filtered to remove noise. Afterwards a linear interpolation algorithm smooth’s the data to further remove erroneous readings, which can then be stored in a database. This database of signal readings is known as the “fingerprint” for the space.

The tracking phase collects a set of signal readings just like in the training phase, however it then compares these readings to those in fingerprint. The closest fingerprint reading is determined and the position in which it was collected during the training phase is used as an estimate to where the device is during the live tracking phase.

A practical problem with this approach is the amount of training time necessary to build a sufficient fingerprint for the tracking phase to work reliably. Many readings would need to be collected in as many different, unique positions in the space as possible. Logistically this would be difficult as the software wouldn’t just need to know the precise positions of all the beacons in the space, but the current position of the device when readings are being taken.

Another problem would be ensuring good coverage over the entire space under many conditions. Changes in temperature and people would affect the RSSI values and so we would need to closely regulate these variables.

Approach Evaluation

In choosing an approach our criteria was as follows:

1. Maximise accuracy.
2. Minimise any on-site setup
3. Minimise development time.

The naïve zone detection approach has the least development time mainly because it’s algorithm only uses basic arithmetic to determine if the device is in a zone or not. As previously explained this approach does not attempt to calculate a position, rather it figures out if enough beacons for a zone can be observed with a close enough proximity. On-site setup time is also fairly minimal as placing a beacon does not need to be in a specific position, just in the area of which a zone is located.

However these 2 simplifications do not maximise accuracy, rather than produce very imprecise results. Fundamentally, position within a space is not being calculated. Simply observing a number of beacons that are close enough does not guarantee the device falls within the boundaries of a zone. The placement of the beacons associated with a zone, and the shape of the zone itself could mean that the device were detected as being inside a zone, when in reality it could just be near to the edge of the zone.

If maximized accuracy alone were important then the fingerprinting approach would be the most suitable. In the case study outlined by Woo et al for construction site in Guangzhou, building a database of filtered and smoothed RSSI

readings to use for comparison with live data yielded positional accuracy within 5m. (This experiment was carried using WiFi access points rather than Low Energy Bluetooth as used in iBeacons.)

Although the fingerprinting approach is promising from an accuracy perspective it has a significantly large overhead. Setting up the construction site for the training phase would involve closely regulating the physical conditions, and it would also necessitate precisely locating a device at many different locations. Also building an interpolation algorithm for filtering RSSI readings along with live comparison component for tracking would take a lot of development time.

Trilateration works out the position of the device as a 2D coordinate in the space. In this regard it is more sophisticated and accurate than the naïve zone detection approach. Also, using trilateration would not involve having a complex training phase as in the fingerprinting approach. Therefore trilateration is the most suitable approach.

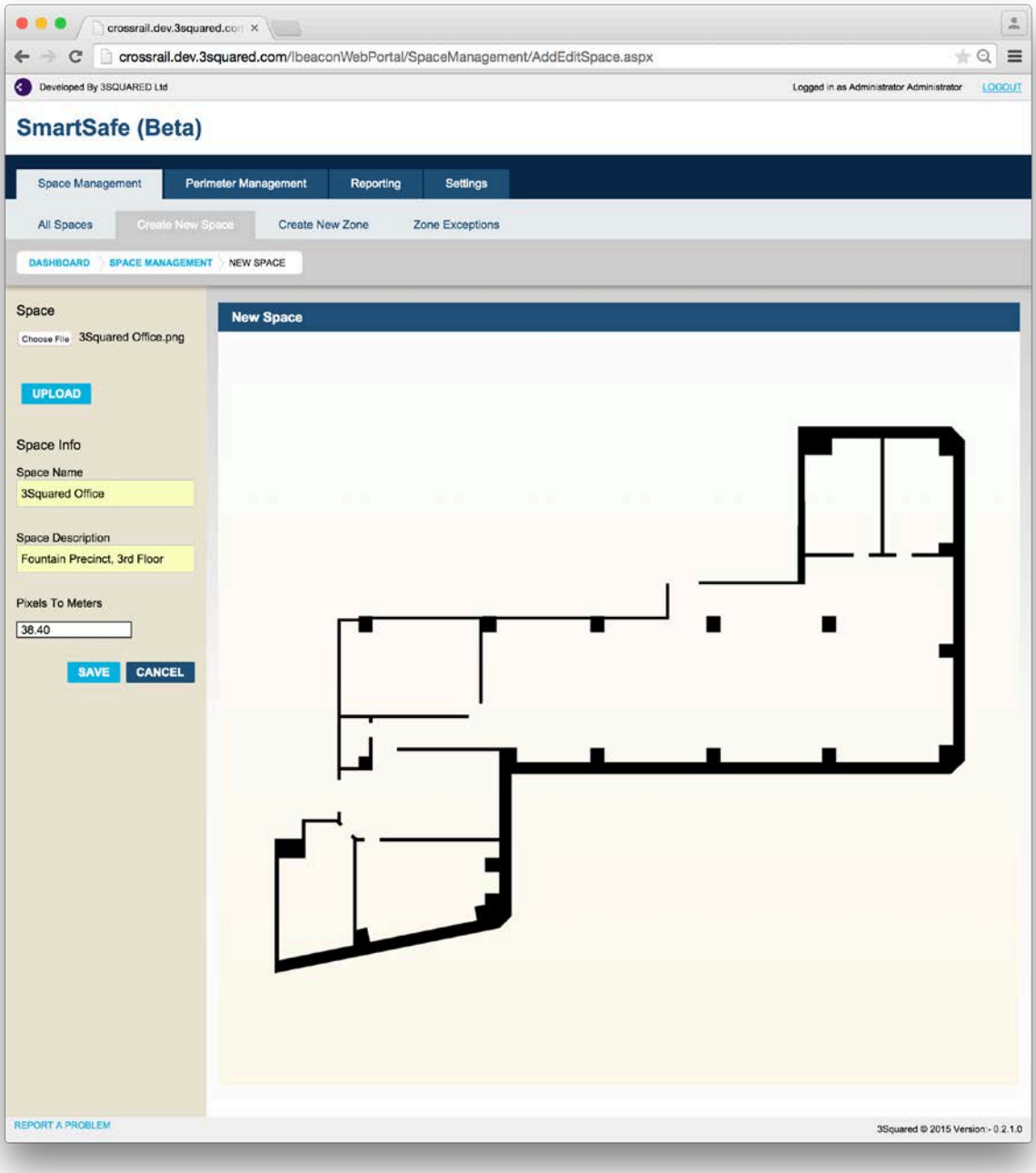
Software

We built two pieces of software that worked together to meet the requirements:

- A web application content management system for managing spaces, zones, beacons, and user permissions.
- A native iPad mobile application that consumed the data from the web application, using it to calculate a user's location within a space and send this information back to the web application for auditing.

Web Application

A user logs into the web application and then creates a space. A space represents the physical area in which the mobile app will be used. An image of the space is uploaded and a name and description added. This image is typically a stripped down floor plan for the space. In the example here an outline of the 3Squared office is used.



The dimensions of the image are used to define the coordinate system for the space. So if an image of 1000 by 1000 pixels is used then the space has coordinate system of 1000 by 1000 points.

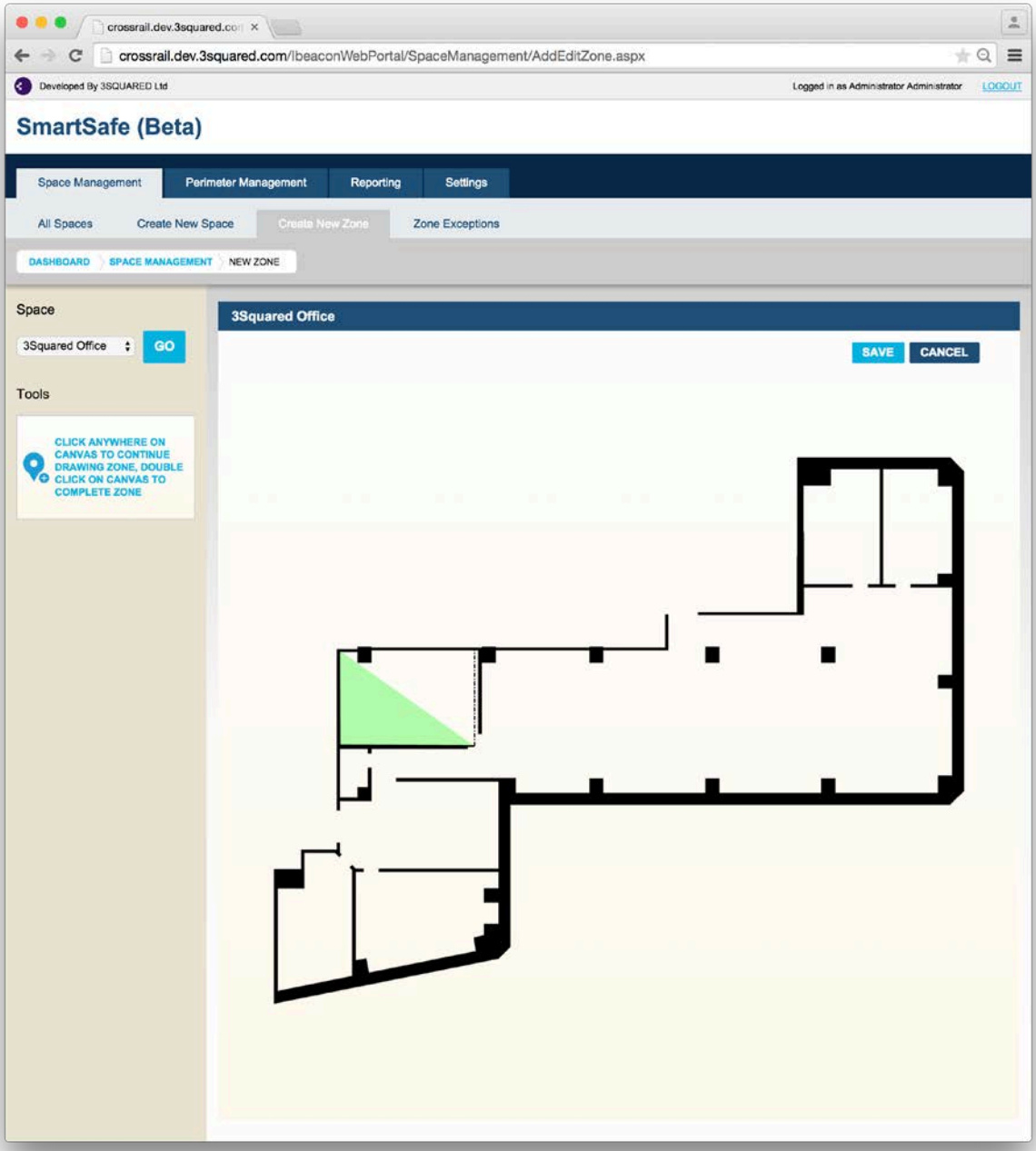
A space also requires a pixels-to-meters value. This will be explained later.

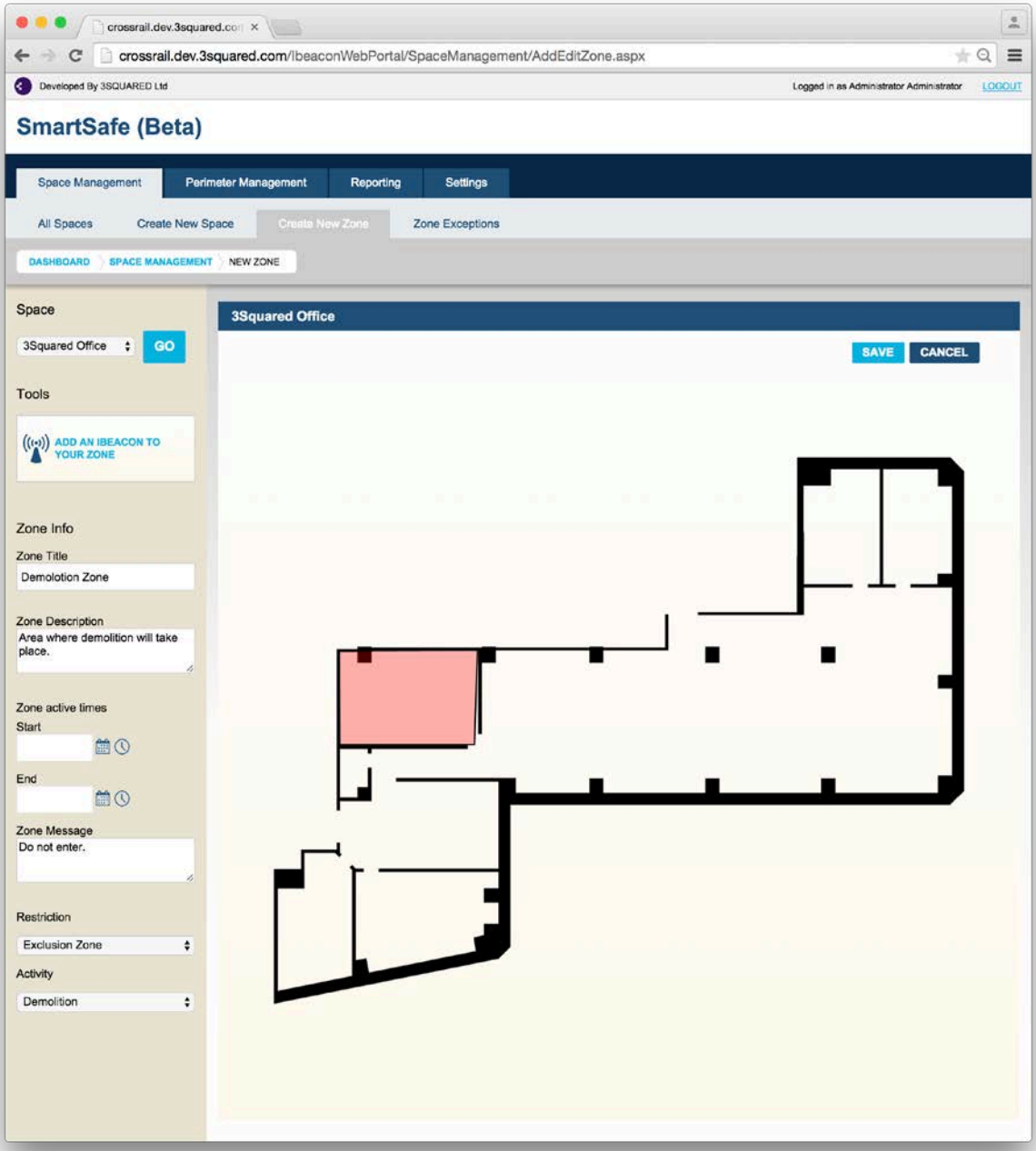
Once a space has been created a zone can be defined within it. A zone's shape can be any straight-lined polygon, which is drawn and positioned on the space by clicking and dragging. Once drawn this polygon is stored as an SVG path. Like a space, a zone is also given a title and description.

In order to restrict access to a zone a restriction type must be selected:

- No restriction safe zone (green): anyone can enter.
- Permit to work zone (yellow): only those with a permit can enter.
- Exclusion zone (red): no one can enter.

A no restriction safe zone defines an area where no activity is taking place and is therefore safe for users to enter. Zones that have either the permit to work or exclusion restrictions define areas where activity is happening and have some form of restriction on who can enter. In both of these cases an activity must be selected for the zone from a predefined list.

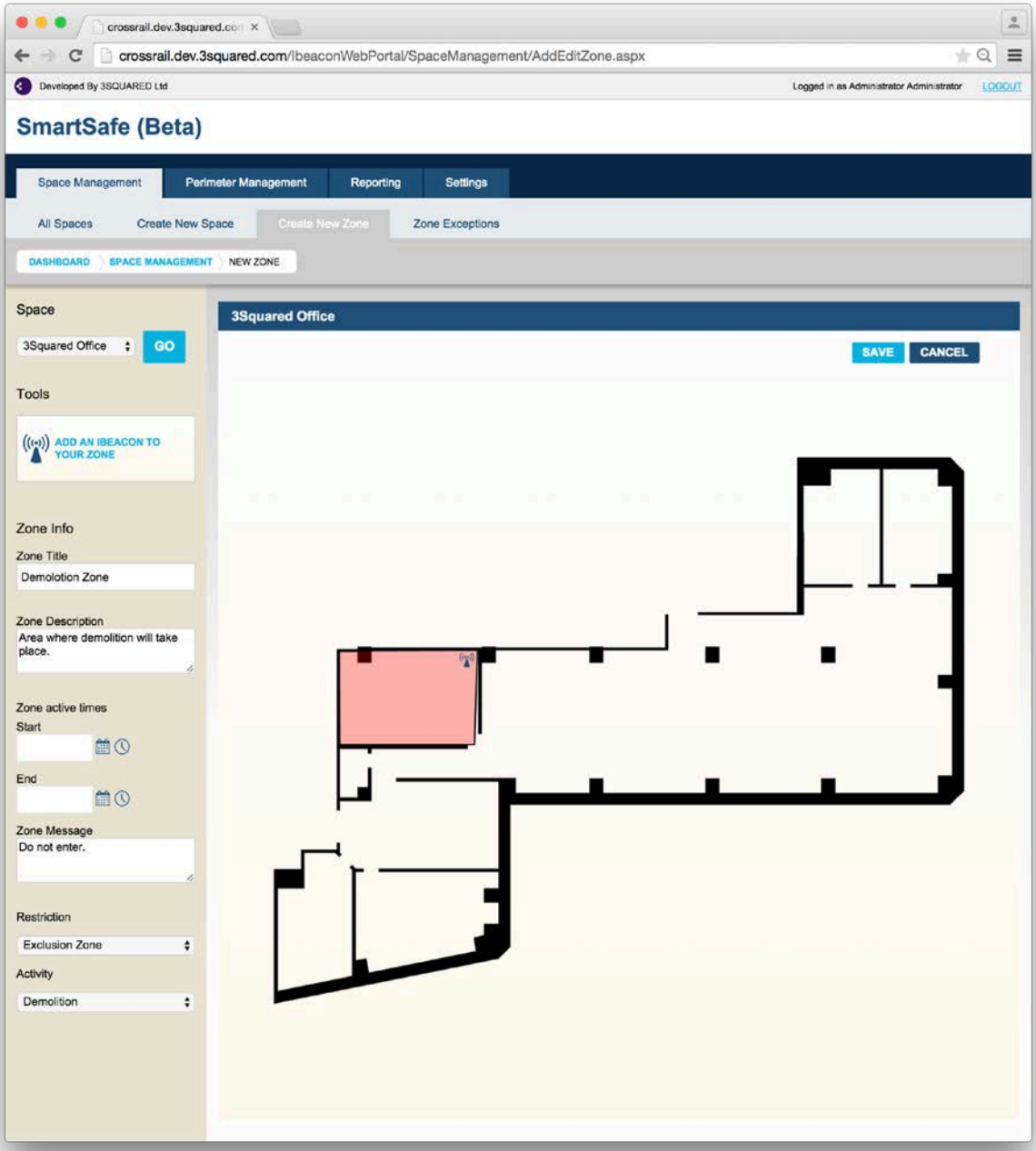




If the permit to work restriction is being used then a set of users with that permit must also be selected. These users are the only individuals who are permitted to enter that zone. Setting permits for users is discussed later.

A start and end time can also be chosen which define when the zone is active. Outside of these hours any permits and restrictions are not enabled, so the zone is effectively a safe zone as any body can enter unchecked.

Beacons are arranged on the space so the mobile device can calculate where it is. Beacons are positioned by dragging them onto the space's floor plan image to a given (x, y) coordinate.

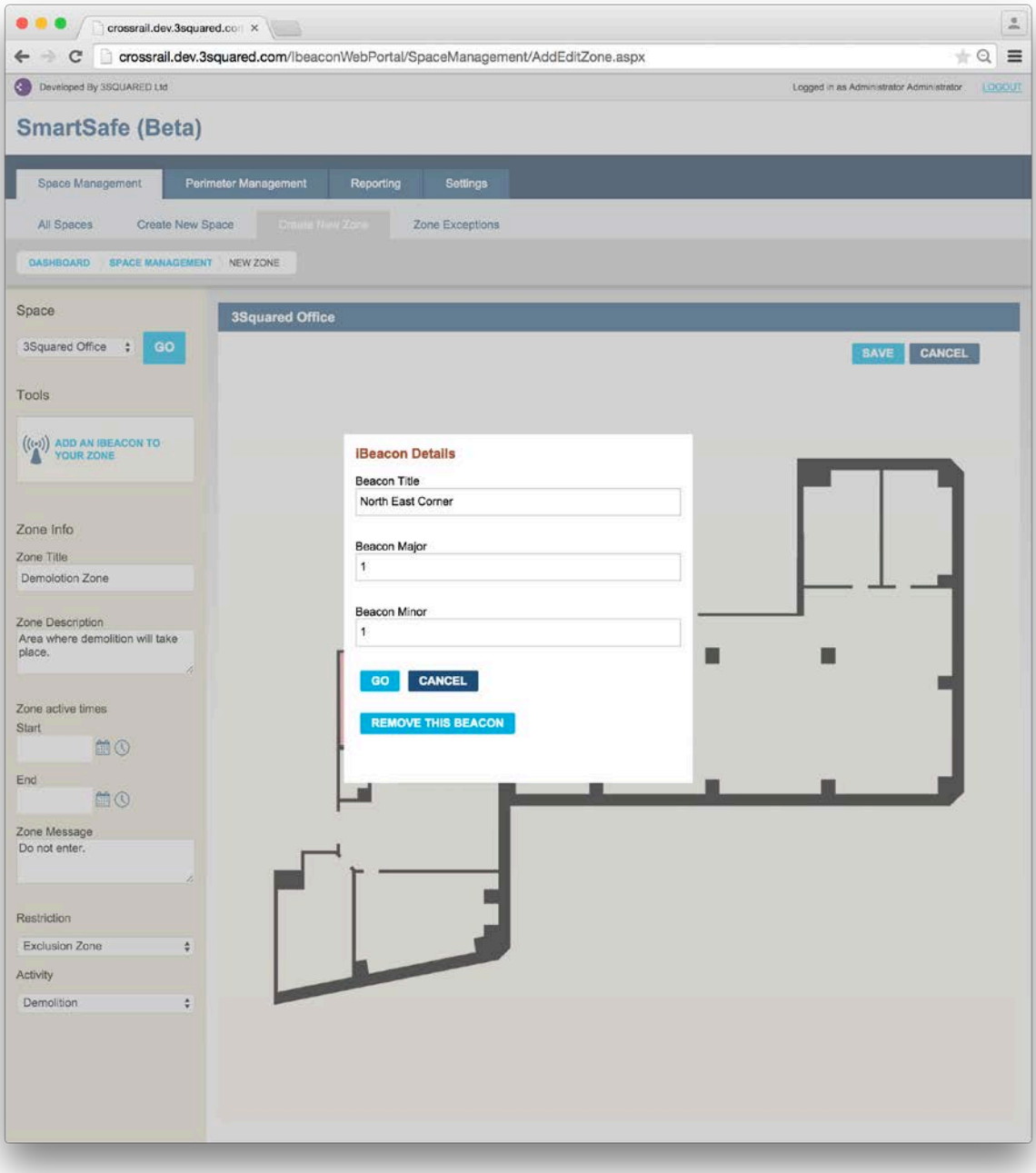


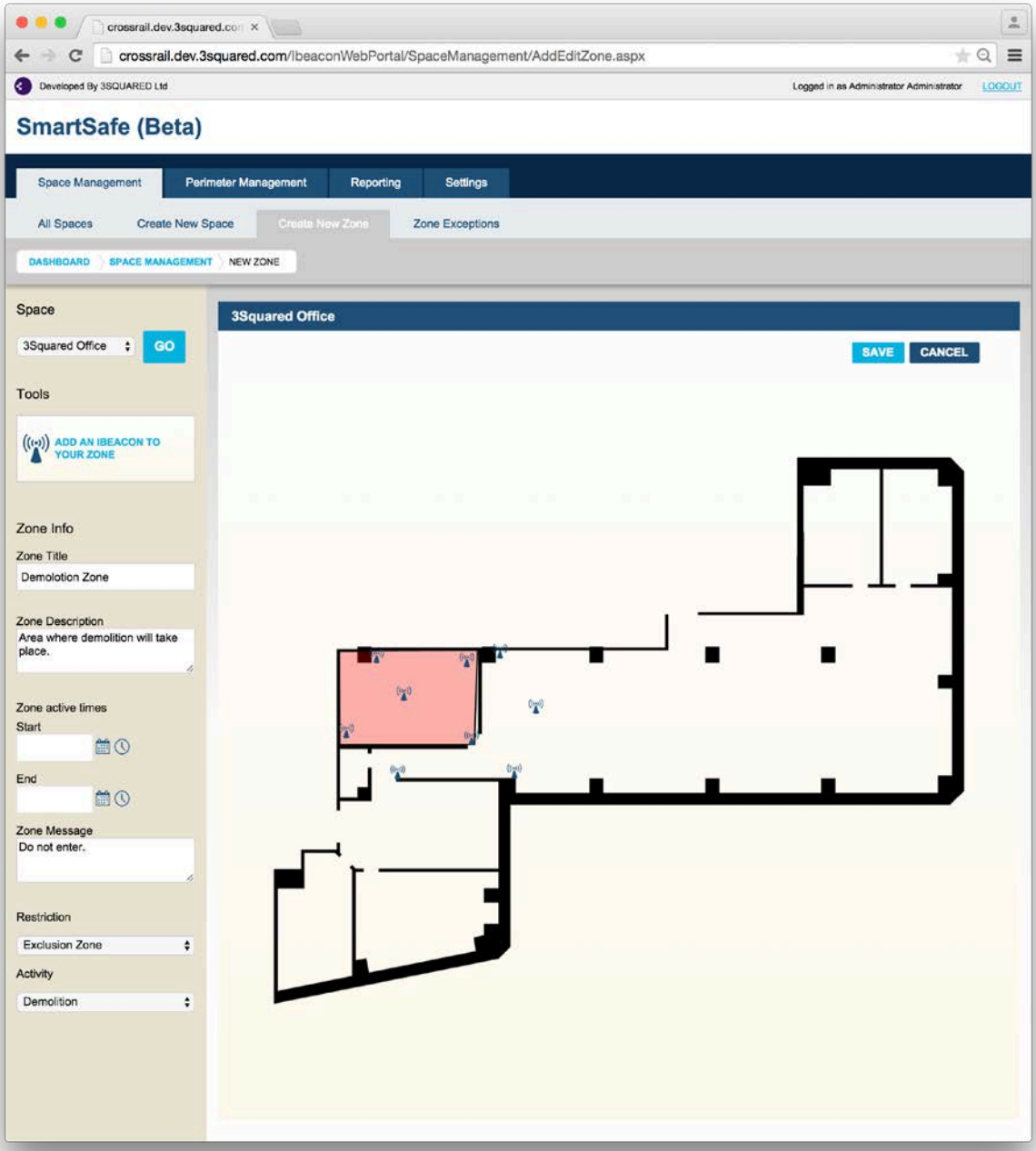
Once placed a beacon must be given 3 values: a major number, a minor number and a name. The major and minor numbers combine to uniquely identify a beacon in a space. The Core Location framework reference documentation defines these numbers as follows:

- *The major property contains a value that can be used to group related sets of beacons. For example, a department store might assign the same major value for all of the beacons on the same floor.*
- *The minor property specifies the individual beacon within a group. For example, for a group of beacons on the same floor of a department store, this value might be assigned to a beacon in a particular section.*

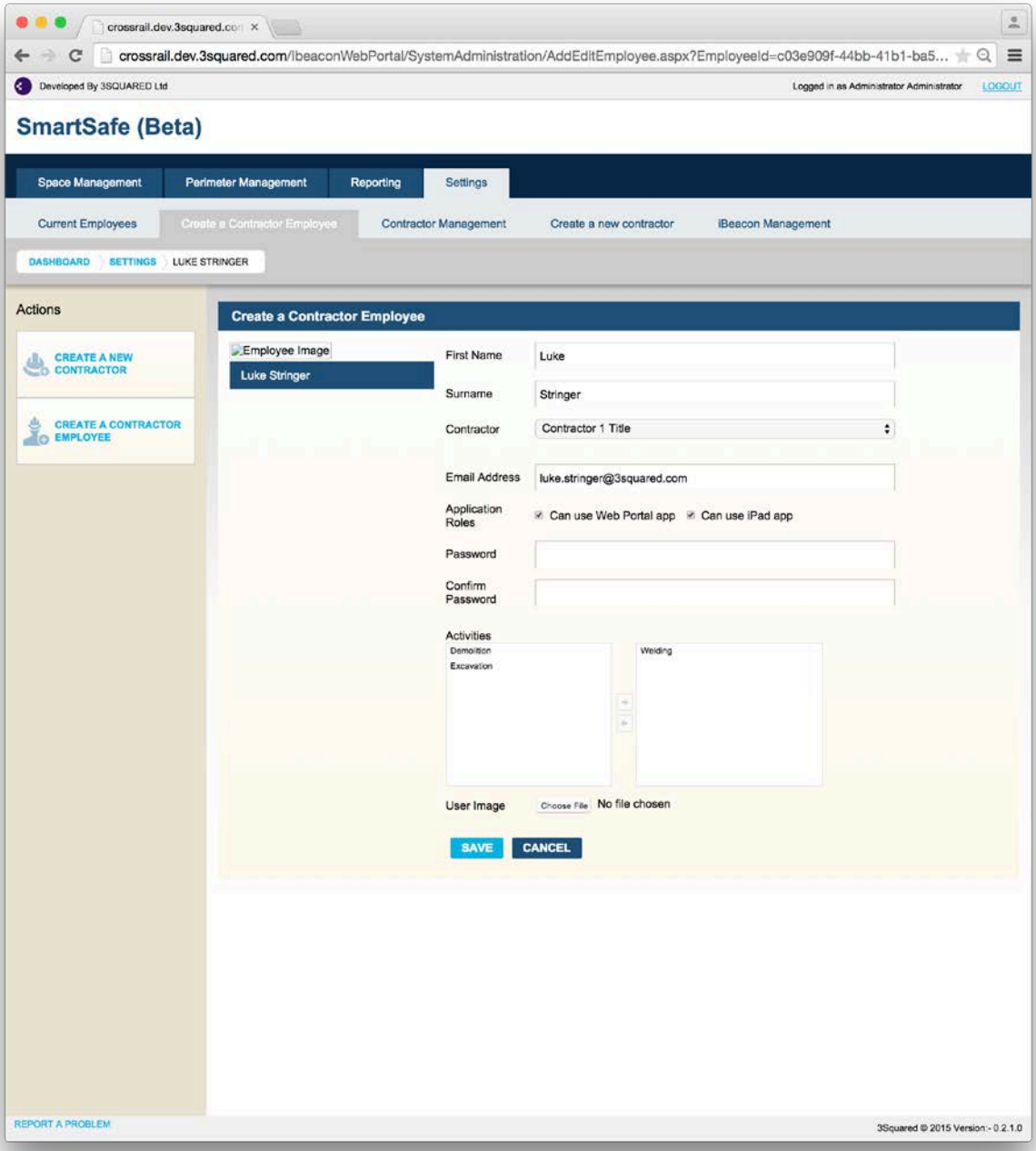
For our purposes is wasn't necessary to have such restrictions for the major and minor numbers, only that a major-minor pair unique identified a beacon. For simplicity this combination was often denoted *a.b*, where *a* was the major number and *b* the minor.

A beacon is also given a user-friendly name. This is not used to uniquely identify a beacon but can help when setting up beacons in the physical environment. For example beacon *1.2* could be given the name *North East Corner*.

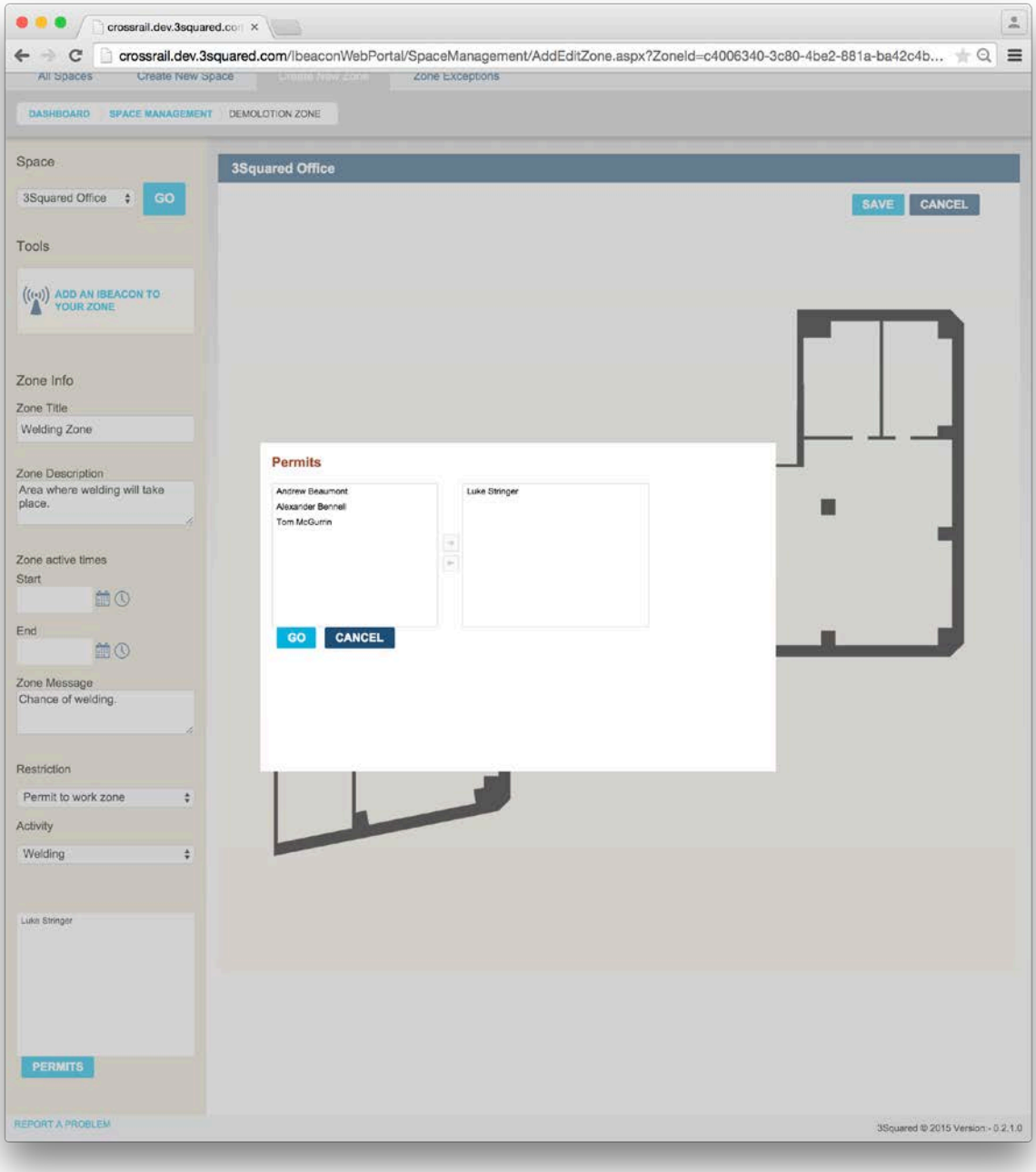




In order to use zones with the permit to work restriction type it is necessary to create and assign permits to users. In the administrative section of the web application a user can be given permission to perform certain activities. Activities a user does not have permission to perform appear in the table on the left, and granting permission moves them to the table on the right.



Once a user has been granted permission for an activity they can be granted a permit to perform that activity from the zone details screen. For example a user may have permission to perform *welding*. If a permit to work zone is setup with the *welding* activity then that user will appear in the list of users who can be granted access to the zone. Note that not all users who have permission to perform welding can enter: only those users who can weld and have been granted a permit to do so in that specific zone can enter.



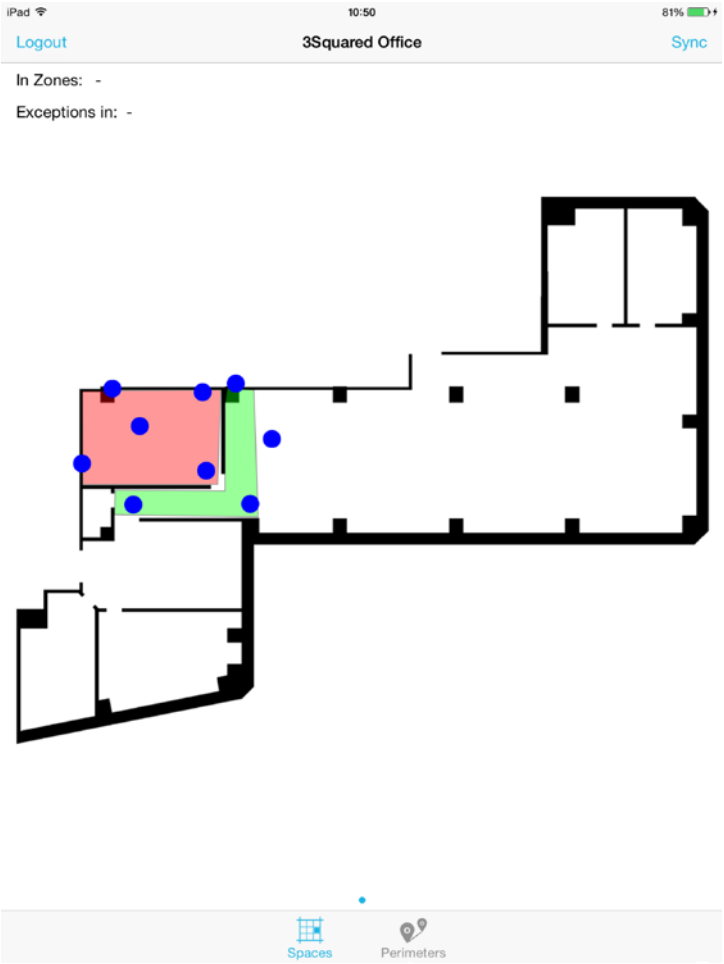
Mobile Application

A user logs into the iPad app with the same details as the web application. The app then requests all the data for that user from the web application via a web API. All the spaces, zones, activities, restrictions, and user permits are downloaded and stored in the app's local database during an initial synchronization process after login.

After data synchronisation a space's floor plan image is rendered on the screen and then zones and beacons are drawn on top. This is accomplished by using the coordinate system of the space defined by the web application. A zone's SVG path can then be used to draw the its shape on top of the floor plan image. Beacon are posited according to their (x, y) coordinates and are also drawn according to the floor plan image. Both zones and beacons are drawn in terms of the

space's dimensions to ensure that what is represented in the mobile app is consistent to what was defined in the web application.

Zones a drawn with a colour corresponding to their restriction type: green for no restriction safe zones, yellow for permit to work zones, and red for exclusion zones. Beacons are coloured blue when the device knows about their existence in the space but has yet to range them via the Core Location API (see below).



Once the iPad knows the coordinates of beacons and their major-minor pair identifiers it can begin to start calculating position.

The Core Location iOS API is used to obtain real-time beacon data. If the user grants access to the device's location then Core Location will provide the app with a list of all the beacons that are in range. The API sorts the beacons by distance from the device and will provide this data no more than once every second for battery considerations.

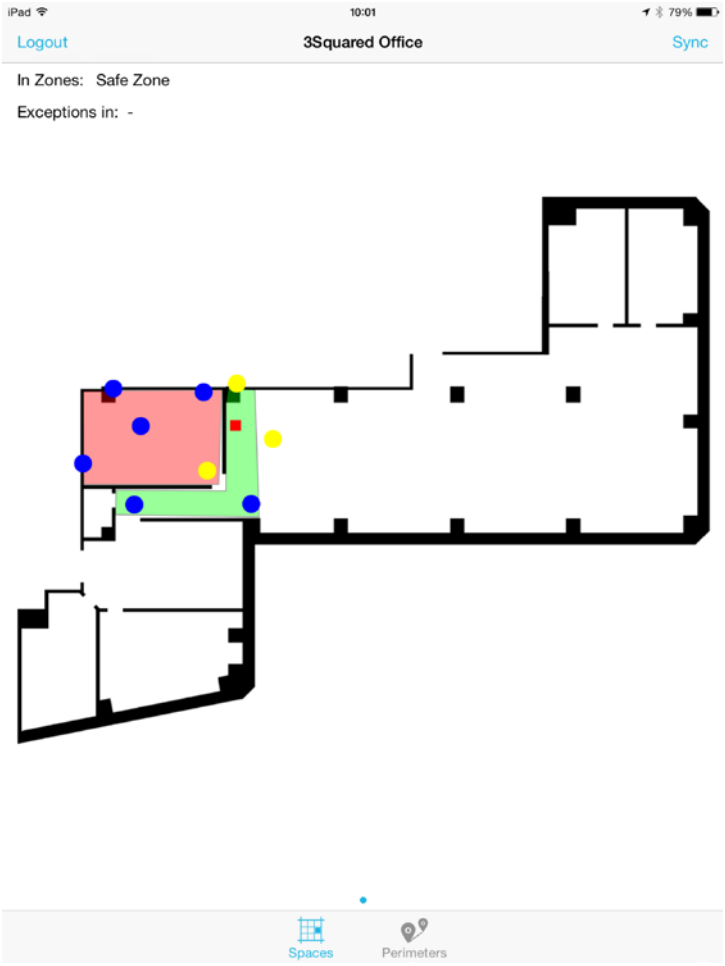
Each ranged beacon has a major-minor pair which uniquely identifies it. A corresponding beacon record in the app's database is searched for that matches this identifier. If such a record is found the app will know the following information for the beacon:

- The major-minor pair identifier.
- The distance to the beacon in meters.
- The (x, y) pixel coordinate of the beacon in the space.

Using the pixels-to-meters value for the space it is possible to convert the distance to the beacon into pixels. When both the distance and coordinate are represented in pixels a circle can be plotted where the center is the coordinate

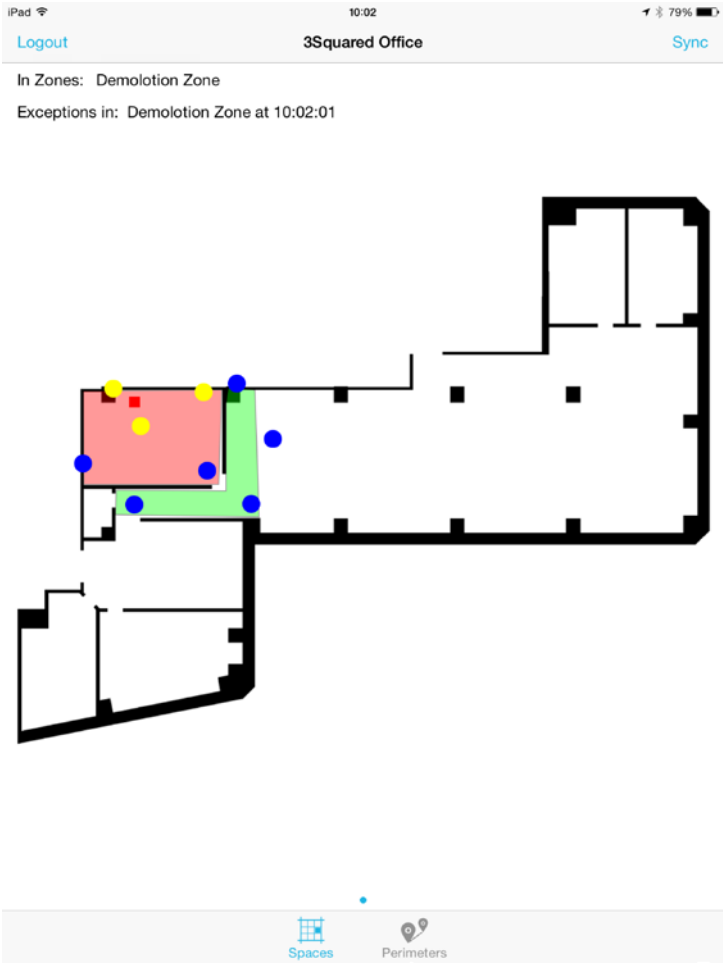
of the beacon and the radius is the distance to the device. The position of the device falls somewhere on the circumference of the circle.

When 3 beacons are ranged then 3 circles can be plotted and position of the device can be calculated using trilateration. To show the user which 3 beacons are being used during the trilateration calculation they are coloured yellow. This differentiates them from the other unused beacons which remain blue.



If Core Location cannot find 3 beacons within range then it is not possible to calculate the device's location. Similarly if 3 beacons are ranged but do not have corresponding entries in the app's database it is therefore not possible to know their coordinates within the space, and again trilateration is not possible.

If the app is able to obtain a coordinate for the device's location then this is drawn onto the space as a red space to provide feedback to the user.



The app then calculates whether this coordinate is inside any of the zones or not. The boundaries of a zone are defined by its SVG path, so if a coordinate is inside these boundaries then the device is inside that zone.

When inside a zone a series of checks is performed to determine if the user is allowed in the zone or not. If the zone has an active period set (the time between the start and end date) but the current time is outside this period then the zone is not active and the user can enter. If the current time falls within this period, or no period was set at all then the algorithm continues and looks at the restriction type. If the device is inside a no restriction safe zone then the user is allowed in. If inside an exclusion zone then the user is not allowed in. If inside a permit to work zone then the algorithm looks at the user's permits. If the user has a permit for the zone's activity then they are allowed in, otherwise they are not.

If it determine that a user is not allowed in a zone, for any of the above reasons, then an exception object is generated and saved to the app's database. This logs the disallowed entry of the user into the zone at the time at which the checks were performed. Exceptions are uploaded to the API to facilitate auditing in the web application.

Testing

In order to test the software 2 kinds of tests were devised. The first looked at the overall functionality of the web and mobile applications and the business logic implementation. The second concentrated specifically on the mobile app and its performance in calculating position.

Business Logic Testing

The following core components of the web application were tested:

- The user authentication mechanism.
- Space Management and its associated actions.
- Employee Management and its associated actions.

The following core components of the mobile application were tested:

- The user authentication mechanism.
- Generating Zone Exceptions.
- The integration between the Mobile and Web Application.

A number of basic scenarios were outlined that tested the business logic:

1. A user entering a zone they do not have permission to access and ensuring an exception is generated.
2. A user entering a zone they do have permission to access and ensuring an exception is not generated.
3. A user entering a zone they previously didn't have access to, but now do, and ensuring an exception is not generated.
4. A user entering a zone they previously had permission to access, but no longer do, and ensuring an exception is generated.
5. A user entering a zone inside it's active period, which they don't have permission to access, and ensuring an exception is generated.
6. A user walking nearby a zone they do not have permission to access, and ensuring an exception was not generated.
7. A user walking on a floor above a zone they do not have permission to access, and ensuring an exception was not generated.

Testing of these scenarios was conducted in 3 environments: 3Squared Ltd's Sheffield office, Crossrail's London Office, and the Crossrail Bond Street construction site. For each scenario a detailed list of steps were devised. First of all these steps setup the data in the web application, then detailed how beacons should be arranged in the test space, and finally set out how the mobile application should be used in order to test the business logic worked correctly.

Tests were initially conducted on a build of the software that used the naïve zone detection approach. As discussed in methodology, this approach does not attempt to calculate a position coordinate for the device. As expected, the test results were unsatisfactory as the software did not reliably detect the user entering or exiting a zone, and therefore exceptions were generated at the wrong times, or not at all when they should have been. Scenarios 1 through 5 that were not consistently fulfilled, and scenarios 6 and 7 were even less successful.

Following the results of this first test it was clear that the positioning algorithm was not fit for purpose. The naïve zone detection approach was simply too unreliable to be used for accurately determining if a user had entered a zone. Work began on implementing a trilateration algorithm in the mobile application ready for a second field test.

The results from the second test were much better. Now the application could calculate a position for the device and determining entry and exit of a zone was much more reliable. Scenarios 1 through 6 passed with a good level of consistency, however it was observed that the performance was best when the device had time to stabiles its readings – that is when moving around the test space the beacon distance readings fluctuated due to signal attenuation. These distance readings (provided from the Core Location API) were used in the trilateration calculation, and so when the device was moving around the position was less accurate. When the device was stationary the distances were more reliable and consistent, and so the position calculated was also more accurate.

Scenario 7 also passed during the second test. When moving to another floor the beacons were not detected, or if they were the algorithm discarded them as the distance readings were greater than what was deemed acceptable for trustworthy data. The proximity value of a ranged beacons was useful here, as when on another floor the app registered beacons with either Far or Unknown proximities and so was able to discard them. The necessary 3 beacons for the trilateration calculation were not available in these circumstances. Therefore entry and exit of zones was not detected, and no incorrect exceptions were generated. The scenario was fulfilled.

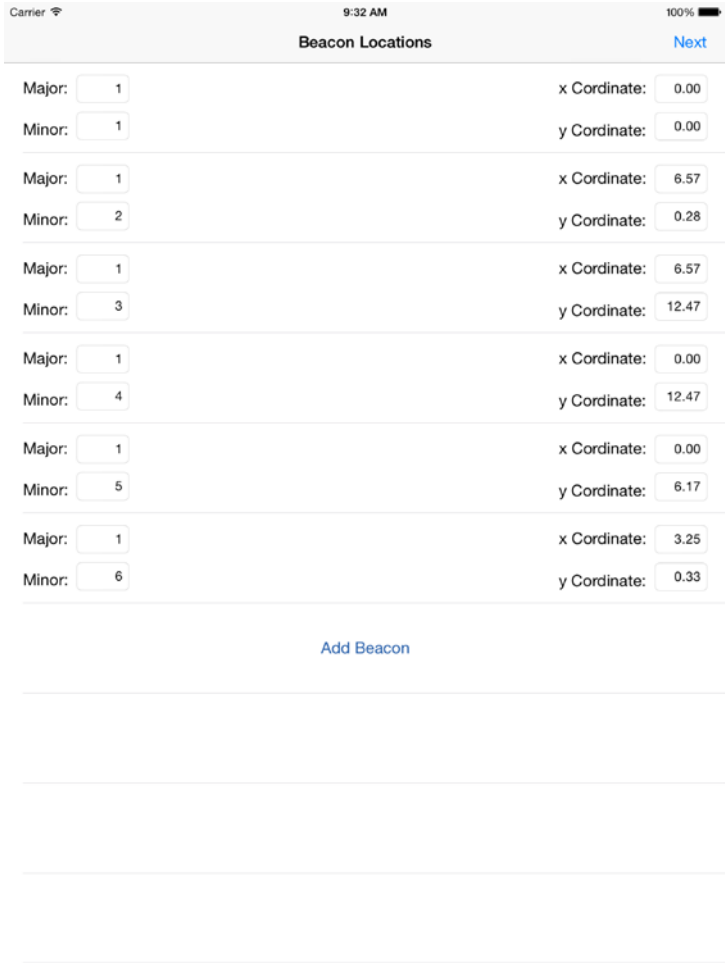
Positioning Testing

In implementing the trilateration algorithm in the mobile app it was necessary to find out more information about how performance was impacted by beacon placement. Although in principle more sophisticated than the naïve zone detection approach, calculating accurate positions using trilateration was still dependent on receiving accurate input data in the first place. This input data was the beacon distance readings from Core Location. How far away beacons could be from a device before their readings became unusable needed to be known. Another mobile application was built to facilitate this investigation.

Unrelated to the web and mobile applications already discussed, this standalone “Data Collection” application provided a way to record known beacon locations in a physical space and then their distance readings as obtained through Core Location from a number of known positioned in the space. By precisely positioning beacons and then the mobile device at known locations it was possible to determine 2 things:

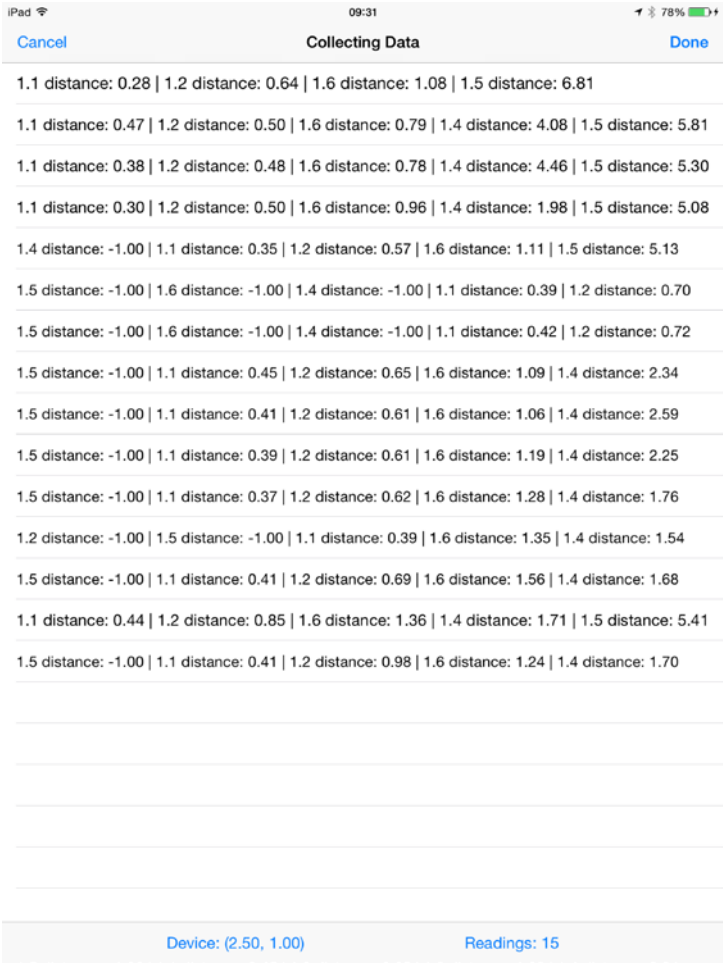
1. How close 3 beacons needed to be for the app’s trilateration algorithm to accurately calculate a position for device.
2. The maximum precision for position calculations that could be expected given a distance from 3 beacons.

The Data Collection app first asked the user to input the known locations of the beacons in the space. For each beacon its major-minor pair identifier and its (x,y) coordinate were needed. The coordinate of each beacon was determine by measuring along the sides of the space where the test was being conducted. An anchor point was designated as coordinate (0,0) and then the beacon coordinates were measured respective to this.



The next step was to position the device at a known location in the space. The (x,y) coordinate was determine using the same method as for the beacons.

Once the app knew where beacons were located and where the device was currently located, the Core Location API was used to get real time beacon distance readings. The readings were collected and saved along with the known beacons locations and the device location (the location at which the reading were recorded). Data collection was then finished and the device could be moved to another known location, which was input into the app. Data collection started once again, and this process continued until a large set of evaluation data had been collected.

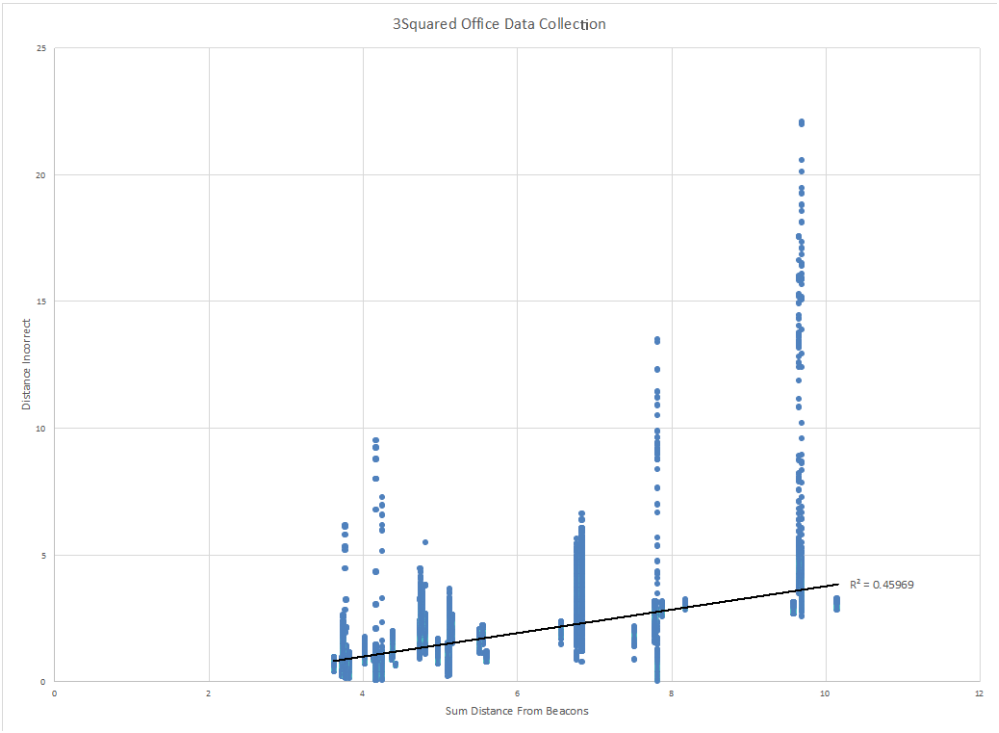


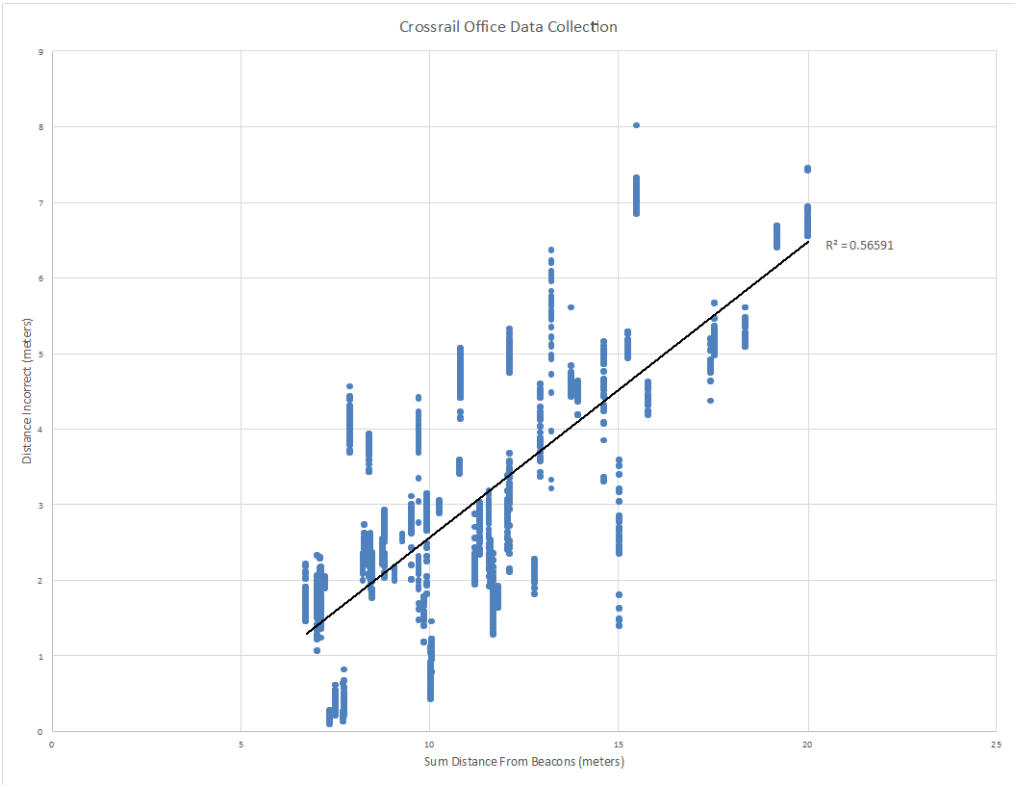
The data collection process was performed at both the 3Squared and Crossrail offices with a variation of beacon numbers and spatial distances. After completion the data was analysed. For each data set all the recorded distance readings were organized into combinations of three. The trilateration algorithm was used with each of these combinations to calculate a position for the device. This calculated coordinate could then be compared to the known correct position to see how well the beacon-distance-combination did.

To compare these three beacon-distance-combinations with each other the sum of the three distances was used as a metric. Therefore the relationship between the sum distance and the inaccuracy of the calculated position could be investigated. The following table shows how these values can be determined for some example data.

Combination	1st Beacon Distance	2nd Beacon Distance	3rd Beacon Distance	Sum Distance	Computed Position	Actual Position	Inaccuracy
1	6.4031	4.1231	5.6568	16.183	(9, 3.9)	(8, 4)	1.000
2	5.831	5.901	5.691	17.423	(5.5, 7.0)	(5, 5)	2.061
3	10.201	20.621	14.424	45.426	(15.9, 19.1)	(20, 20)	4.196

The sum distance for each of the combinations was calculated along with the inaccuracy in the position trilateration calculation. These were then plotted as two graphs.





During the data collection tests at the 3Squared office 4 beacons were used, and a total of 5821 readings were taken at 10 different device locations, resulting in 21125 usable combinations. During the Crossrail office tests 9 beacons were used and a total of 728 readings were taken at 7 different device locations, resulting in 3064 usable combinations. As more beacons were used in the second of these tests the beacons were position at greater distances from each other so that a larger spread could be recorded.

The data collected at 3Squared offers insight into when a device can be located quite near to beacons – where the sum distance from 3 beacons used for trilateration is around 4 to 10 meters. The analysis has shown that under these circumstances an inaccuracy of no more than 5 meters can be expected most of the time.

The data collected at Crossrail offers insight into when a device is located further away from beacons – where the sum distance is higher at around 5 to 20 meters. Under these circumstances it appears that the positional inaccuracy is similar, with an expected margin of error of no more than 7 meters.

It was therefore determined that in order to achieve positional accuracy to within 5 meters using a trilateration approach the mobile device should be no further than a sum distance of around 10 to 15 meters from 3 beacons: no more than 5 meters from each beacon.

By decreasing the sum distance the precision in the positional calculation could be increased. This could be accomplished by increasing the density of beacons with the test space. Put simply: the closer a device is to the to beacons the more reliable the distance readings are, and the more reliable the positional calculation is.

Findings

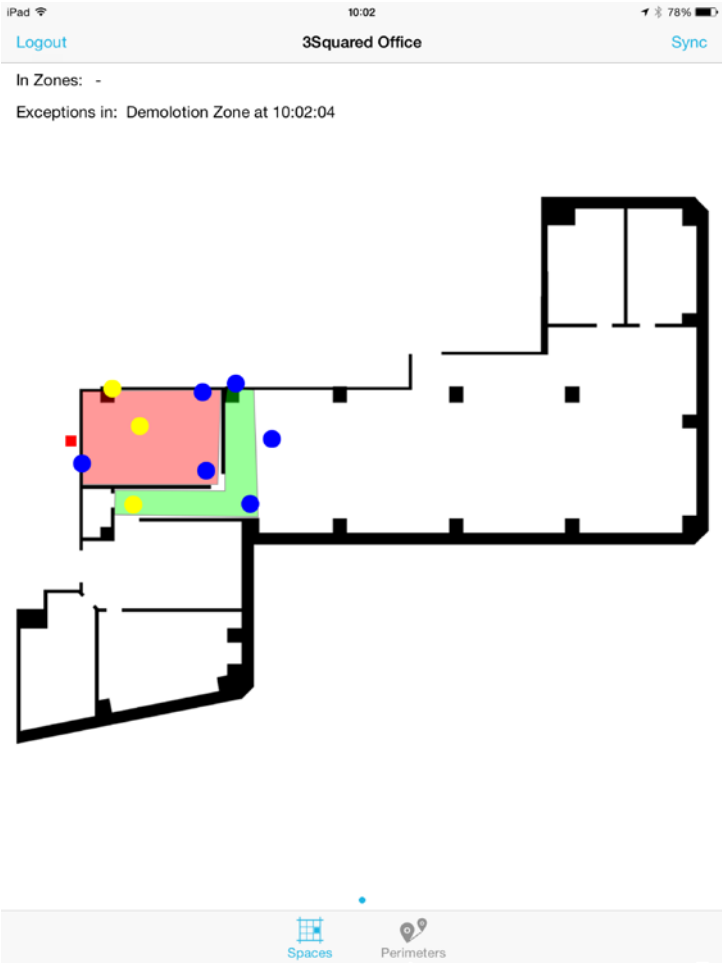
From testing the application in both an office and construction site environment it was observed that correctly placing beacons was crucial in getting the best precision. Beacon locations in the software were represented as 2 dimensional coordinates without a concept of altitude. Therefore it was left to the user to determine the best vertical placement for beacons – either on the floor on somewhere higher on a wall. The Estimote beacons that were used had an adhesive material on the back so it was possible to stick them to a wall or vertical surface.

We found that when beacons were placed directly onto the floor the application found it very difficult to get reliable readings in order to calculate a position. From the user's point of the view it appeared as though the software had frozen, however the application was simply unable to get 3 good enough beacon distance readings for trilateration. When 3 readings could be used this occurred so in frequently that the application was unable to detect entry and exit of zones in real time, rendering the application unusable.

When beacons were placed higher up by attaching them to walls performance was greatly improved. The mobile application was able to find 3 reliable distance readings almost all of the time. This meant that the application updated every second and calculated a new position for the device.

It was also observed that the position of beacons in relation to zones was also important for maximising the software's ability to accurately detect zone entry and exit. When creating a zone in the web application if beacons were placed both inside and outside of the zone the mobile app was able to get reliable distance readings around the boundary of a zone. By having reliable data at the boundary, accurate detection of entry and exist was possible. When beacons were simply placed inside the zone and/or along the perimeter this kind of data was not available, so position calculations were less precise.

Occasionally the calculated device location would change dramatically within a few seconds. This was manifested as the device location identifier on screen jumping around. This would happen when moving around the space when signal attenuation was at its worst. When this happened the location reading would sometimes incorrectly position the device inside a zone, resulting in the business logic being executed and potentially an incorrect exception being generated. Other times the location that was calculated was actually impossible for the device to be – outside of the office space for example. In both these examples that application would need improvement.



Another result of the position calculation being erratic was that multiple exceptions could be generated when only a single one was correct. For example when inside a zone but near to the boundary the location could be calculated as being inside on one reading, then outside on the next, and then back inside on the next. As position calculation was only accurate to within 5 meters zone detection near to the boundary could be particularly unreliable. If the user was not permitted to enter the zone multiple exceptions could be generated as the application (incorrectly) calculated them entering, exiting, and then re-entering.

Conclusions & Recommendations

Web Application

Testing showed that the web application worked well to configure the data necessary for the mobile application to work. The different configuration options for spaces, zones, beacons, users and permits fulfilled requirements 1, 2, and 5. Configuration data was successfully communicated to the mobile device, which in turn successfully sent back generated exceptions when users entered zones that they were not allowed to enter. These exception reports fulfilled requirements 8 and 20.

Mobile Device Management

Unfortunately it was not possible to fulfil requirements 3 and 4 which aimed to restrict device capabilities. As discussed in the Methodology section, although an MDM solution could be used to restrict what a user can and cannot do in real-time, the lack of a programmable interface to send out these restrictions based on a user's location (as

calculated via trilateration) means that the requirements were impossible to implement. Furthermore, the inability to guarantee Wi-Fi connectively on a construction site also means that an MDM solution would not be possible as configuration settings would need to be pushed to a device in real time.

A recommendation here would be to monitor the MDM software market to see if future products provide a way to programmatically create and distribute configuration settings based on some specific criteria – for our requirements, a user entering or existing a zone.

Location Calculation

The mobile application's performance in calculating location and communicating this back to the web application fulfilled requirements 6 and 7. Location calculation was to a reasonable level of efficiency - if the iPad was never more than 5 meters away from 3 beacons, position could be calculated to within 5 meters of accuracy most of the time. Ultimately however precision to within 5 meters is just not good enough for a safety critical application where the margin between a safe a hazardous zone is often only centimetres. To ensure safety an alternative approach to the trilateration model should be investigated. From our research a Finger Printing approach (as described in Methodology) would be a good candidate as although this involves a lot more setup and greater complexity to the software, better accuracy and precision could be obtained.

However if trilateration was kept as the core algorithm for location calculation then one recommendation for future work would be to investigate how erroneous zone detection could be decreased. On the boundaries of a zone it was particularly noticeable that the mobile application was unreliable in detecting whether the device was inside or outside of a zone. Work could be undertaken to try and make this less of a problem by setting a "minimum occurrence threshold" for zone detection. For example, entry to a zone could become dependent on 3 consecutive readings that position the device inside the zone. Therefore, if the position calculation was erratic (inside a zone, then outside, then back inside) incorrect zone detection would not be a problem as the threshold would not be met. This logic could also be applied to leaving a zone, where 3 consecutive readings would be necessary to detection exiting. The minimum occurrence threshold could be investigated to determine the optimal value through further field testing.

Another way to improve correct zone detection would be to provide extra context to the space during creation to designate areas that are impossible for the device to actually be in. In the same way zones are drawn on a space through clicking and dragging, inaccessible areas could be defined in the same way. If the mobile application calculated a location inside one of these areas, then the calculation would simply be discarded. This would resolve the problem of incorrect zone exit detection because the device was calculated as being outside the zone but in an inaccessible area.

When beacon distance readings become erratic taking an average over a period of time could also prove useful. The Core Location API provides beacon readings every second. Rather than taking the three best beacons per second and using them for position calculation, an average of the distances could be taken over a period of seconds, and then the best three beacons could be selected. This would smooth the data and mitigate against erroneous readings throwing the precision out. An investigation into the optimal window for averaging could be undertaken, and more a sophisticated weighted averaging approaches could be looked into.

Finally, the second generation of beacon technology could be investigated. It would also be recommended that more research be undertaken into the products developed by Aruba and Meridan. Their solution shows impressive results however it is unclear to what extent it could be used in a safety critical application.

Ultimately however as BLE beacons and the Core Location API are fundamentally not designed to provide accurate distance readings to a mobile device, we feel that another technology should be investigated instead. We recommend that a fingerprinting approach using Wi-Fi would be the best avenue for this investigation. Perhaps a combination of Wi-Fi nodes and BLE beacons could be used to create the digital fingerprint.

References

http://www.crossrail.co.uk/benefits/innovation/

http://meridianapps.com/sdks/

http://techcrunch.com/2014/11/04/how-the-49ers-are-using-beacons-to-help-you-find-hot-dogs-and-beer/

http://www.arubanetworks.com/solutions/mobile-engagement/

http://nfarina.com/post/101309491728/lets-talk-about-beacons

http://www.absolutemanage.com/en/manage/mdm

https://developer.apple.com/library/prerelease/ios/documentation/CoreLocation/Reference/CLBeacon_class/index.html

Indoor Position Detection Using WiFi and Trilateration Technique - Nor Aida Mahiddin, Noaizan Safie, Elissa Nadia, Suhailan Safei, Engku Fadzli.
Faculty of Informatics, University Sultan ZainalAbidin, Gong Badak Campus, Terengganu, Malaysia

Indoor Location Using Trilateration Characteristics - B Cook (1, 2, 3), G Buckberry (2), I Scowcroft (2), J Mitchell (1), T Allen (3)
1: University College London
2: Siemens Communications
3: Nottingham Trent University

Application of WiFi-based indoor positioning system for labor tracking at construction sites: A case study in Guangzhou MTR - Sunkyu Woo (a), Seongsu Jeong (a), Esmond Mok (b), Linyuan Xia (c), Changsu Choi (d), Muwook Pyeon (e), Joon Heo (a)
a: Department of Civil and Environmental Engineering, Yonsei University, Seoul 120-749, Republic of Korea
b: Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong, PR China
c: School of Geographical Science and Planning, Sun Yat-sen University, Guangzhou, PR China
d: Communication LAB., Central R&D Center, LS Industrial System Co., Ltd., Republic of Korea
e: School of Civil Engineering, Konkuk University, Seoul, Republic of Korea