

Imperial College London  
Department of Computing

# Modelling Infection Spread Using Location Tracking

by

Andrew Mason

Submitted in partial fulfilment of the requirements for the MSC Degree  
in Computing Science of Imperial College London

September 2008



## **Abstract**

Location tracking technology has improved greatly over the last few decades, to the extent where it is now possible to record the location of an object to within 25cm of its actual location. New applications of this technology are constantly being discovered, from stock tagging to improving a football team's tactics. My project focuses on healthcare applications, specifically on the spread of infection.

My aim is to show that by tracking the locations of individuals in a closed environment, it is possible to record the nature and frequency of interactions between them. Further, that it is possible to use this data to predict the way in which an infection will spread throughout such a population given certain parameters about it, such as its contact and recovery rates.

In this project, I present a software package that I have designed that is capable of recording and then replaying location data provided by the Ubisense Location Tracking System. I describe how the software employs a combination of SIR modelling and the epidemiological technique of contact tracing in order to predict the spread of an infection. Finally, I use the software to run a number of experiments using a sample data set, and compare the SIR graphs generated from these to similar graphs generated using the traditional SIR differential equations.



# Acknowledgements

I would like to thank the following people for their help and support throughout this project:

My supervisor, William Knottenbelt, who has remained enthusiastic and helpful at all times, and consistently replied to emails promptly no matter what time I sent them.

My girlfriend, Rachel D'oliveiro, and indeed everyone who has put up with me going on about location tracking over the past few months.

Tom Oliver, and everyone else who has given me an excuse to have a coffee break.

John Gibbon, for teaching me about SIR models in M2A1. Never did I imagine that those notes would ever be useful again...



# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Motivation . . . . .	9
1.2	Aims and Objectives . . . . .	10
1.3	Report Structure . . . . .	11
<b>2</b>	<b>Background</b>	<b>13</b>
2.1	Epidemiology . . . . .	13
2.1.1	The SIR model . . . . .	13
2.1.2	Contact Tracing . . . . .	14
2.2	The Ubisense Location Tracking System . . . . .	16
2.2.1	Overview . . . . .	16
2.2.2	Technology . . . . .	16
2.2.3	Hardware . . . . .	16
2.2.4	Software . . . . .	17
<b>3</b>	<b>The Software</b>	<b>19</b>
3.1	Requirements . . . . .	19
3.1.1	Evaluation of location tracking hardware . . . . .	19
3.1.2	Analysis of Infection using location data . . . . .	19
3.2	Design . . . . .	20
3.2.1	Programming Languages . . . . .	20
3.2.2	Program Design . . . . .	21
3.3	Implementation . . . . .	22
3.3.1	Overall Architecture . . . . .	22
3.3.2	Layout . . . . .	23
3.3.3	Functionality . . . . .	25
<b>4</b>	<b>Testing</b>	<b>31</b>
4.1	The Ubisense System . . . . .	31
4.2	The Software . . . . .	32
4.2.1	Infection radius . . . . .	33
4.2.2	Infection Persistence . . . . .	35
4.2.3	Contact rate . . . . .	36
4.2.4	Recovery Rate . . . . .	36

<b>5</b>	<b>Evaluation</b>	<b>41</b>
5.1	The Ubisense System . . . . .	41
5.1.1	Setup . . . . .	41
5.1.2	Operation . . . . .	41
5.2	The Software . . . . .	42
<b>6</b>	<b>Conclusion</b>	<b>45</b>
6.1	Overview . . . . .	45
6.2	Obstacles . . . . .	45
6.3	Future Work . . . . .	46
<b>A</b>	<b>User Guide</b>	<b>51</b>
A.1	Introduction . . . . .	51
A.2	Requirements . . . . .	51
A.3	Getting Started . . . . .	52
A.4	Modes of Operation . . . . .	53
A.4.1	Record . . . . .	53
A.4.2	Playback . . . . .	53
<b>B</b>	<b>Class Diagram</b>	<b>57</b>
<b>C</b>	<b>Ubisense accuracy data</b>	<b>59</b>
<b>D</b>	<b>Sotware testing data</b>	<b>61</b>
D.1	Infection radius . . . . .	62
D.2	Infection persistence . . . . .	64
D.3	Contact rate . . . . .	68
D.4	Recovery rate . . . . .	70



# List of Figures

2.1	A typical SIR model. Red represents susceptible individuals, green infectious and blue recovered . . . . .	14
2.2	An example of a graph produced by contact tracing. Each node represents an infected individual, and each line connects two individuals who have been in contact . . . . .	15
2.3	The compact Ubitag (left), and slim Ubitag . . . . .	16
2.4	The Ubisense Location Engine Configuration Program . . . . .	17
3.1	A class diagram for the software. To aid readability, functions and variables governing layout have been omitted. A full class listing can be found in Appendix B . . . . .	21
3.2	Overall architecture of the project . . . . .	22
3.3	A simple map and database insert/retrieval tool . . . . .	23
3.4	A later version of the application . . . . .	23
3.5	The susceptible tag status window . . . . .	24
3.6	The time bar, and play and record buttons . . . . .	24
3.7	The Infection source box and tag status windows . . . . .	24
3.8	Green dots representing an infectious area centered on Server 3, whose name is coloured green to indicate that he is infectious . . . . .	25
3.9	The <i>Open</i> Dialog Box . . . . .	26
3.10	Infection Source Dialog Box . . . . .	27
3.11	Algorithm to determine infection status . . . . .	28
3.12	An SIR-style graph generated by the software. Red: susceptible, Green: infectious, Blue: Recovered . . . . .	29
4.1	Accuracy analysis - red dots denote survey positions . . . . .	31
4.2	A screen capture showing locations from the sample data set . . . . .	32
4.3	SIR graph for the control experiment (red: susceptible, green: infectious, blue: recovered) . . . . .	33
4.4	SIR graphs for increasing infection radius. Red: susceptible, Green: infectious, Blue: recovered . . . . .	34
4.5	SIR graphs for increasing infection persistence. Red: susceptible, Green: infectious, Blue: recovered . . . . .	35
4.6	SIR graphs for increasing contact rate. . . . .	37
4.7	SIR graphs for increasing recovery time. . . . .	38
A.1	A screenshot of the Control Panel, showing the different sections . . . . .	52
A.2	The <i>Open</i> Dialog Box . . . . .	54

A.3	The <i>Infection Source</i> Dialog Box . . . . .	55
A.4	The graph window . . . . .	56
B.1	A full class listing, generated by Microsoft Visual Studio. Grey attributes relate to layout settings, and some included methods are no longer used by the current version of the software. . . . .	58

# Chapter 1

## Introduction

### 1.1 Motivation

The ability to accurately track the location of an individual has improved greatly over the last thirty years. From the launch of the GPS system in 1978 [9], through wireless technologies such as 802.11a/b/g and bluetooth, and technologies such as assisted GPS, the accuracy of real-time location tracking systems that are widely available today is sub-10m. While this is useful, and widely utilised, for many location-based applications, the ability to track individuals with an accuracy greater than this has only recently been developed. Increasingly, technologies such as RFID [6] and ultra-wideband [1] are being exploited to provide accurate location tracking of a large number of individuals.

Many uses of this sort of technology are already apparent, for example keeping track of workers in a dangerous environment to protect their safety [6], and tracking football players to provide post-match analysis [1]. However, there are undoubtedly many more to be discovered as the technology matures and is more widely available. One such application is in the healthcare system. Location tracking systems are already being used to keep tabs on at-risk patients, who need constant monitoring to protect their safety [2]. Another use may well be in the tracking and prevention of infections, and it is this application that this project aims to explore.

Many diseases are passed from person to person by physical encounters - from intimate contact in the case of sexually transmitted infections, to simply being in the near vicinity in the case of more virulent airborne infections. While the former type of encounter is relatively easy to keep track of, until recently the latter has been almost impossible to study. With the ability to track individuals' locations to a sub-1 metre accuracy, though, this may change.

If we assume that it is possible to track the spread of an infection at the scale proposed, there are many possible applications of such a system with regard to the prevention and further understanding of epidemics. For example, during a real epidemic, information about contact between individuals

could give doctors access to a list of people ordered by the probability that they are infected. This would allow informed targeting of treatment, and quarantine if necessary. If it were possible to alter the model in real-time to reflect the infection status of patients, the model would become more accurate as time went on, and could retro-actively be updated to calculate new lists as required. If recorded, such information would also be invaluable in predicting the spread of future infections.

In a similar vein, given a relatively predictable system (such as a hospital), it should be possible to determine which individuals have the most infection-spreading contact with others. This would again allow doctors to create a list of people for whom immunisation would be a priority in the event of an epidemic. This could dramatically reduce its spread, as those responsible for spreading the infection to most people would no longer do so.

Further applications are also possible. Given information about the people who are infected, and when they showed symptoms, it might be possible to determine the likely source of an infection within a closed system (again, such as a hospital). From this point, a more traditional contact tracing study could be performed to identify those at risk in the outside world.

There are doubtless many more applications of such information, which extend far beyond the scope of this project. The motivation for this project, therefore, is the desire to enable the possibility of collecting such data, so that its applications can be further explored.

## 1.2 Aims and Objectives

The broad aim of this project is to investigate the feasibility of using real-time location tracking as a basis to create models that are useful in the study of epidemiology. This can be broken down into the following parts:

- Evaluation of location tracking hardware

The location tracking hardware used for the project is relatively new technology, and to my knowledge has never been used for the kind of study that I am proposing. Therefore it is necessary to carry out an assessment of the hardware in order to determine both its accuracy and reliability. If either of these are not sufficient, there is very little chance that the data obtained using the hardware will be useful in any kind of infection model.

- Suitability of location tracking data for analysis of infection spread

Even assuming that the data collected is both accurate and reliable, it does not follow that it will be of any use in modelling a real-world infection. Infection modelling is rarely attempted at this level, and little or no research regarding it exists. I therefore aim to compare the results of the data I obtain to a simple infection model that is in widespread use, and see if there is a correlation.

- Investigation of possible applications of location tracking data

Assuming that accurate and reliable data can be collected, and that it is suitable for use in studying the spread of infection, there are undoubtedly many specific applications, some of which are outlined in the previous section. By collecting some sample data, it should be possible to explore some of these, and to obtain some results. Specifically, I aim to carry out an analysis of how the immunisation of different individuals affects the spread of a given infection in a given situation. I also aim to analyse a sample set of data in order to determine the order in which individuals should be immunised in order to reduce the spread of infection.

In order to achieve these objectives, my aim is to design a software package that can record location tracking data, and allow the user to analyse the data in a variety of ways in order to evaluate its effectiveness in the analysis of infection. This will involve being able to create a scenario in which a given set of individuals is infected, and viewing the results of this as contacts between individuals occur in the monitored environment. Ideally, some analysis of the results will be possible, so that it can be compared to existing epidemiology models.

### 1.3 Report Structure

A brief introduction to epidemiology, and more specifically the SIR model and contact tracing, can be found in section 2.1. From here, I look at the Ubisense System in section 2.2, and consider how it might be applied to the methods mentioned in the previous section, and the ways in which it might be a useful tool in the study of epidemiology. I also briefly discuss the technology itself, and how it provides highly accurate location tracking data.

The main part of the report (chapter 3) focuses on the software application that I designed to test the functionality of the Ubisense System, and to try to apply it to various toy-models of epidemics. I describe the technologies and languages used, and justify the design decisions that led to the finished program. I also describe the main functionality of the program.

Chapter 4 contains the results of a number of experiments that I performed in order to test both the Ubisense hardware and the software that I designed. I obtain quantitative data regarding the spread of infection throughout a given scenario, and provide a qualitative comparison with an existing infection model.

The operation of the program and the Ubisense system itself is contained in chapter 5. Here, I examine the performance of the system, and whether it might be useable in a real-world study of an epidemic.

Finally, chapter 6 outlines the main conclusions of the project, and goes into some detail about what I have learned from the project, and the directions in which I think further work might go. Due to the short time scale of the project, there were inevitably some omissions and bugs that I didn't have time to deal with, and these are also listed here.



# Chapter 2

## Background

### 2.1 Epidemiology

Epidemiology, or the “study of the distribution and determinants of health-related states or events in specified populations” [8], is concerned with the spread of disease and its effect on people. This in itself encompasses a range of disciplines, from biology to sociology and philosophy, all of which are utilised to better understand and contain the spread of infection.

#### 2.1.1 The SIR model

One of the most basic models that is used in epidemiology is the SIR model. This was introduced as far back as 1927 by Kermack and McKendrick [7], and although simple, is a good model for many infectious diseases. The SIR model assumes that a given fixed population whose members are at risk of a given infection can be split into three groups:

- $S(t)$  Susceptible (those at risk of infection)
- $I(t)$  Infectious (those who have been infected by the disease, and can infect others)
- $R(t)$  Recovered (those who have recovered from the disease, and therefore are no longer at risk)

As denoted above,  $S$ ,  $I$  and  $R$  are all functions of time, and vary as an epidemic progresses. Assuming the total population remains constant (and is denoted by  $N$ ), it is clear that the following holds:

$$S(t) + I(t) + R(t) = N$$

Using this equation as a base, we then consider the transition rates between the three states, i.e. the rate at which susceptibles become infected, and at which those who are infected recover. These are denoted by  $\beta I$  and  $\nu I$  respectively, where  $\beta$  is the contact rate (the probability of a susceptible individual becoming infected given an encounter with an infectious individual), and  $\nu$  is the rate at which those who are infected recover. Thus it is possible to derive a set of differential equations to describe the system:

- $\frac{dS}{dt} = -\beta IS$
- $\frac{dI}{dt} = \beta IS - \nu I$
- $\frac{dR}{dt} = \nu I$

While this system cannot be solved analytically, plotting it on a graph yields a typical timeline for an epidemic, for example in figure 2.1<sup>1</sup>.

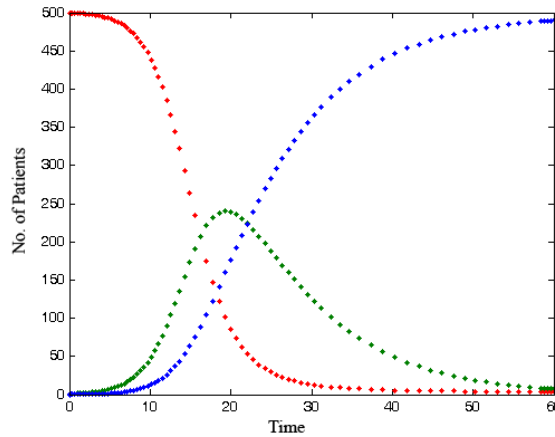


Figure 2.1: A typical SIR model. Red represents susceptible individuals, green infectious and blue recovered

In the real world, of course, the system is much more complicated. Indeed, the model can be extended to include many more factors, for example an incubation period. However, at a basic level the SIR model provides a reasonable approximation for the spread of a disease, and therefore I shall use it as a reference point. If my modelling program produces infection models that are similar to real life, it is likely that the results of the two will show a high correlation.

### 2.1.2 Contact Tracing

A disease spread model is a good analytical tool, but by their very nature, models are only an approximation of real life. In order to evaluate a model, we need to collect empirical data regarding the transmission of a disease. Contact tracing is a method used by epidemiologists that involves creating a list of infection-spreading contact that one individual has had with another [5]. For

<sup>1</sup><http://en.wikipedia.org/wiki/Image:Sirsys-p9.png>



example, assume individual  $A$  has an infection that is spread by physical contact. Person  $A$  recalls that he has recently come into contact with persons  $B$  and  $C$ .  $B$  recalls that since coming into contact with  $A$ , he has come into contact with person  $D$ , and so on. Using this method, it is possible to build up a graph (for example figure 2.2<sup>2</sup>) of those who may have become infected, complete with the probability of infection if  $\beta$ , the contact rate, is known.

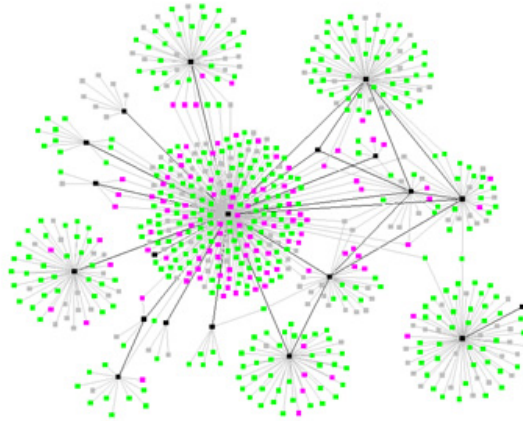


Figure 2.2: An example of a graph produced by contact tracing. Each node represents an infected individual, and each line connects two individuals who have been in contact

In certain cases, this method works well. It is particularly useful in the study of sexually transmitted infections [4] (STIs), when infection-spreading encounters are likely to be remembered, and the timing pinpointed relatively accurately. However, if person  $B$  can become infected by person  $A$  simply by being within a given distance of  $A$  for a given time, it is obvious that the method breaks down. Short of requiring every individual in a given environment to record every such encounter, it is impossible to create such a graph.

However, with recent advances in location tracking technology, it has become possible to record the location of any number of individuals reliably to a reasonable accuracy. This in turn makes it possible to create such a graph, even when the briefest of encounters can be the cause of an infection being transmitted.

My project aims to combine the two techniques described above. Using the location tracking data, it should be possible to construct an SIR model for a given infection whose parameters are known by performing a contact tracing study. Conversely, it should also be possible to determine factors (such as the contact rate) about a given infection if the time of infection is known.

---

<sup>2</sup>[http://www.visualcomplexity.com/vc/images/28\\_big01.jpg](http://www.visualcomplexity.com/vc/images/28_big01.jpg)

## 2.2 The Ubisense Location Tracking System

### 2.2.1 Overview

The Ubisense location tracking system claims to allow a user to monitor the position of a number of individuals to a sub-15cm accuracy, and with a sub-second update frequency. Each individual carries a small tag, which is tracked by a network of sensors. The position of each tag is then relayed to a server running the location platform software, and can be visualised on screen.

### 2.2.2 Technology

The Ubisense system uses ultra-wideband (UWB) technology to track each tag's position. Each tag sends out UWB pulses, which are received by the sensors. Each sensor records the signal's time and angle of arrival. This data is collated by the master sensor for a given cell, which then calculates the tag's position as a set of x, y and z coordinates, and records the standard error of the location, and the time it was received. The location data is then transmitted to the server over the network.

The system also utilises more conventional radio signals to transmit other information, such as button press events that are generated when one of a tag's buttons is pressed. In order to maximise battery life, a tag will go to sleep when it is stationary, and begin transmitting again when it starts to move.

While ultra-wideband pulses are capable of transmitting large amounts of information using very little power, it is worth noting that UWB pulses cannot travel through water. This means that the human body (which consists mostly of water) can block the signal.

### 2.2.3 Hardware



Figure 2.3: The compact Ubitag (left), and slim Ubitag

Each Ubisense tag, or 'Ubitag'(figure 2.3<sup>3</sup>) resembles a pager in size and shape, and can be carried easily by an individual. This means that his or her location can be accurately tracked with little

---

<sup>3</sup>Taken from the Ubisense training slides, courtesy of Ubisense

or no inconvenience. Due to the limitations of UWB mentioned above, however, the use of tags in a location where there a large number of people could be problematic. Each tag also has a number of buttons, which can be monitored through the Ubisense API, and used to signal events.

The sensors must be placed so that each tag to be tracked is in direct line of sight of at least two of them for the majority of the time. In practice, this means that a 10m x 5m room requires a minimum of four sensors to provide adequate coverage. Once the sensors have been placed, it is necessary to calibrate them. This involves measuring their exact location in the room, and then using a tag to collect enough location data for the software to calculate each sensor's pitch, yaw and roll, and the length of ethernet cable (the timing cable) between it and the master sensor. This process is fairly lengthy, but once done does not need to be repeated unless the sensors are moved.

## 2.2.4 Software

The Ubisense package includes several pieces of supporting software. At its core is the Location Engine Configuration program, which allows the user to calibrate the sensors and set parameters such as the dimesions of the room. it is also possible to assign names to the different tags, in order to differentiate between them.

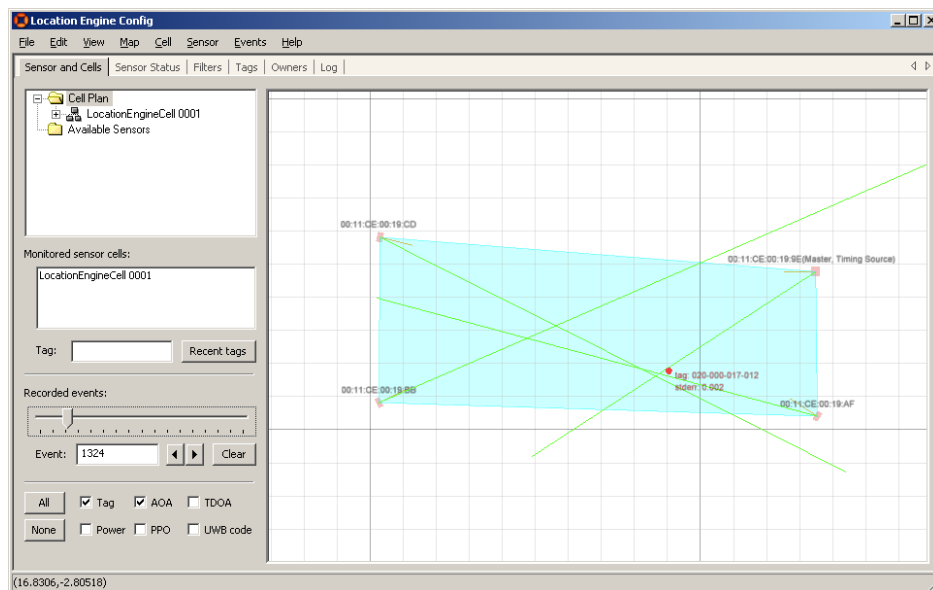


Figure 2.4: The Ubisense Location Engine Configuration Program

Also included in the package is a mapping application, which is used to add detail such as walls and tables to the system; a simulator, which allows a user to simulate the motion of the tags by defining 'scripts'; and various other applications that are used to control the services that run on the server and client computers.

The Ubisense software comes complete with a fully-featured .NET API, which allows access to a large proportion of the settings and data. Location data within the system is held in schemas, which can be thought of as tables. As an example, the locations of each tag are held in such a

schema, in which each tag has a row containing its name, co-ordinates, time, and standard error of the last reading. Using the API, these schemas can either be polled, or an event handler can be set up that is called whenever a change is made, or a new row inserted.

# Chapter 3

## The Software

### 3.1 Requirements

#### 3.1.1 Evaluation of location tracking hardware

The software must be able to receive data from the Ubisense location platform, and translate it into a form that is understandable by the user. Essentially, this means that at any given point, a list of tags that are being tracked must be available, accompanied by their positions. Ideally, they will be displayed on some kind of map to aid visualisation. If this is accomplished, it will be possible to visually estimate the accuracy of the hardware, and view any bugs or flaws that it may have when reporting data.

If data is recorded, it can be compared to measured positions to provide a more quantitative analysis of the system's accuracy.

#### 3.1.2 Analysis of Infection using location data

Using a set of recorded data, or a live stream, it must be possible to apply a model of an infection to the scenario. Using the SIR model described in the introduction, this means that it must be possible to determine which tags are susceptible, infectious and recovered, and to allow for the spread of infection from infectious to susceptible individuals according to the contact rate. It must also be possible to view the results of such an analysis, and to view the effects of an infection over a given period.

The user should be given the maximum flexibility to define different variables for each session, including where any sources of infection are, the contact rate of the infection, the times at which individuals are immunised or become infected etc.

There should be some way of visualising a session to allow the user to see more easily what is going on.

To summarise, therefore, the finished program should have the following features:

- Ability to receive and process data from the Ubisense system regarding the location of tags.
- A record function to store the location data so that it can be analysed or played back later.
- Some sort of map or other visualisation that allows users to see where tags are at a given time.
- A comprehensive set of controls that allows the user to determine and view variables relating to a recorded session. These should include:
  - Whether each tag is susceptible, infectious or recovered at any given time.
  - The addition of sources of infection
  - The ability to set parameters regarding the type of infection, for example the contact rate and recovery rate.
- Some way of viewing the results of a session in an easily readable form, that can be compared to existing models.

## 3.2 Design

### 3.2.1 Programming Languages

The Ubisense Location Engine is intended for use on a Microsoft Windows platform, with its most extensive API available for C#. This, combined with the graphical nature of the program requirements (the need to visualise data and results, etc) means that the most obvious language choice for designing the package is C#, within the .NET framework.

Given the sub-second frequency of the location data, it is clear that a large amount of information will need to be stored for even a relatively short session. A quick estimate reveals that for eight individuals, given a session length of 20 minutes and an update frequency of 4 points per second, 38400 individual points will need to be recorded. In order to manage such a large volume of data, I decided to use a database, and chose Microsoft's SQL server for its easy integration with C#. However, in practice any SQL-compliant database would suffice.

### 3.2.2 Program Design

Although the software package is intended largely as a proof-of-concept, to show whether it is possible to use real-time location tracking in the useful analysis of infectious diseases, there is the possibility that it could be extended to provide a useable tool at some point in the future. Additionally, it is much easier to demonstrate new technology to those who do not necessarily have an in-depth understanding of the processes behind it if the interface is clear and intuitive. Therefore, this was my main aim in designing the layout. My intention was to make it easy to view tag locations, and simple to view and alter the infection statuses of tags.

The large amount of information and number of elements that needed to be dealt with in the program meant that a good object-oriented design was important. Figure 3.1 shows the class diagram for the program, and illustrates how different elements are managed. The control panel is the main form, and maintains lists of tags and infection sources. These each have their own methods, including a draw method and one to update their locations, which are in themselves objects.

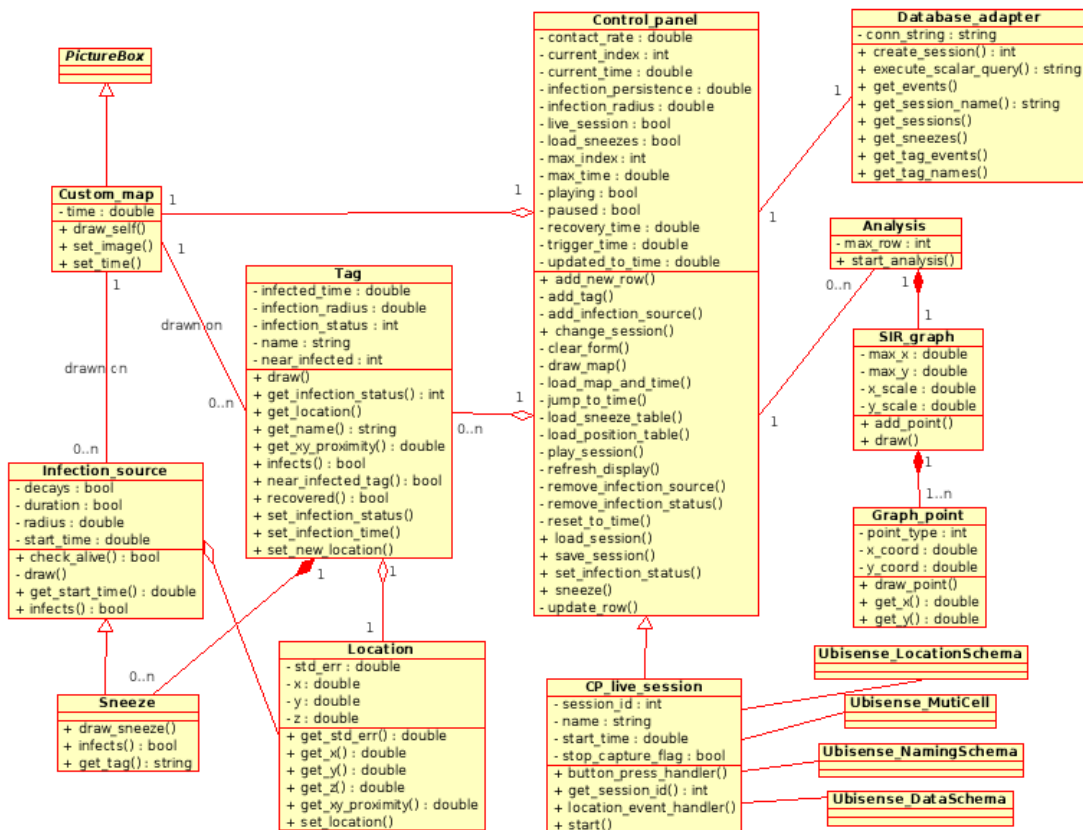


Figure 3.1: A class diagram for the software. To aid readability, functions and variables governing layout have been omitted. A full class listing can be found in Appendix B

The result of this design is that the program is clearly laid out. In addition, the low coupling that is maintained by each object having its own methods means that the program is easily extensible, and new objects can be added as necessary without extensive re-writing.

## 3.3 Implementation

### 3.3.1 Overall Architecture

The Ubisense system communicates with the central platform server over a Power-Over-Ethernet (POE) switch, which provides power as well as communication to the sensors. Any number of client machines running the Ubisense Service Controller Software can be connected to the network, and receive updates from the server via multicast IP packets. The overall top-level architecture can be seen in figure 3.2. The client machine receives the location updates, which are passed through the .NET API into the software. The information is then displayed in the application, or can be saved to a database.

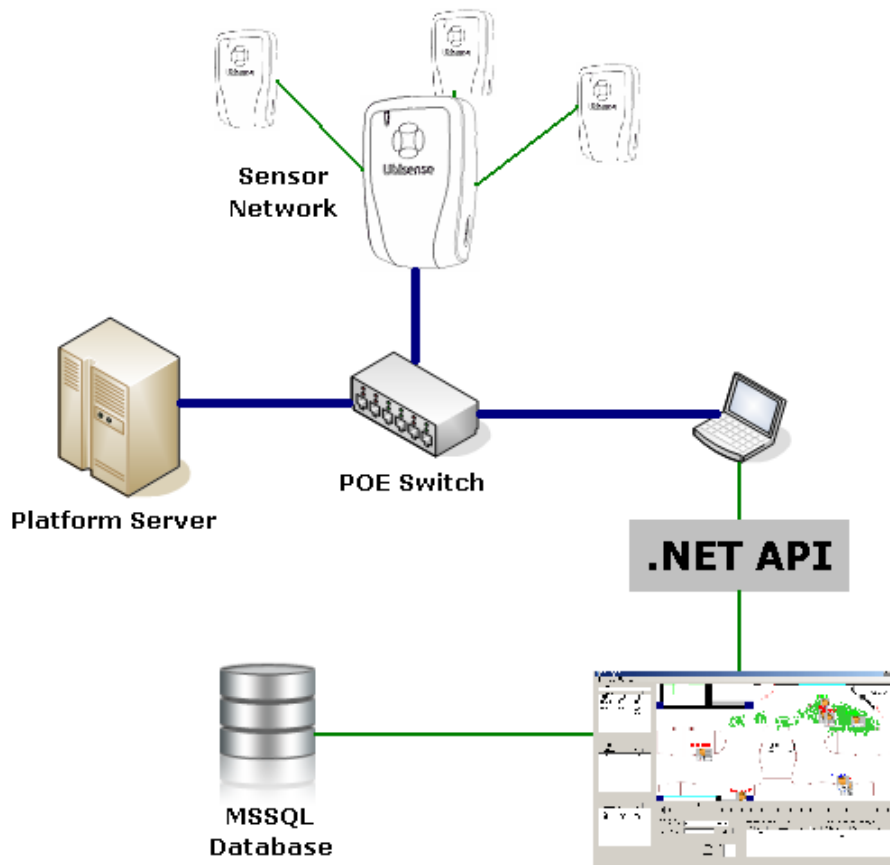


Figure 3.2: Overall architecture of the project

In practice, the GUI went through several iterations. The first was a simple application (figure 3.3) designed to show the location of a number of tags at a given point. This was to ensure that I was using the Ubisense API correctly, and that the hardware was providing a reasonable representation of the tag locations. When this was functioning adequately, I then moved on to saving and retrieving information from the database.

The database contains one table that contains details of sessions (a unique session id and an



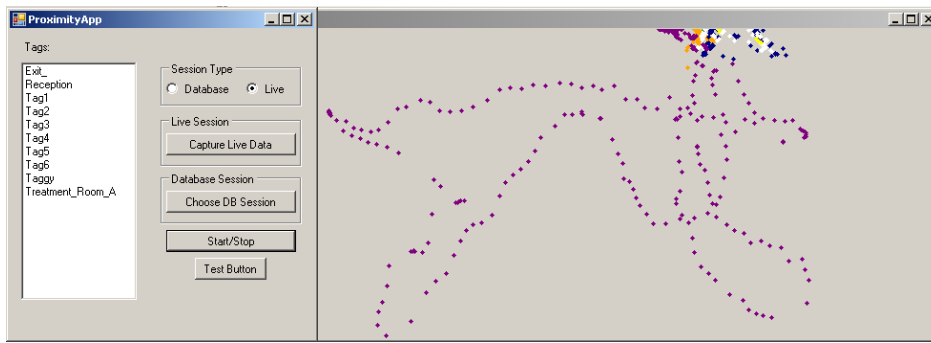


Figure 3.3: A simple map and database insert/retrieval tool

identifying session name), and one table that contains all of the session events. Each session event row contains the session id it relates to, a unique tag name, the x, y and z co-ordinates of the tag, and the standard error and time of the reading. When a session is being recorded, each location received by the program is translated into a row of this format which can later be saved to the database. This allows for easy playback of the session, as each row is read in turn and translated into a visual representation on the map.

The simple map from the test application became the focal point of the final application. Microsoft's GDI proved adequate for displaying the map, and it evolved to include more details, notably labels to allow a user to differentiate between tags.

### 3.3.2 Layout

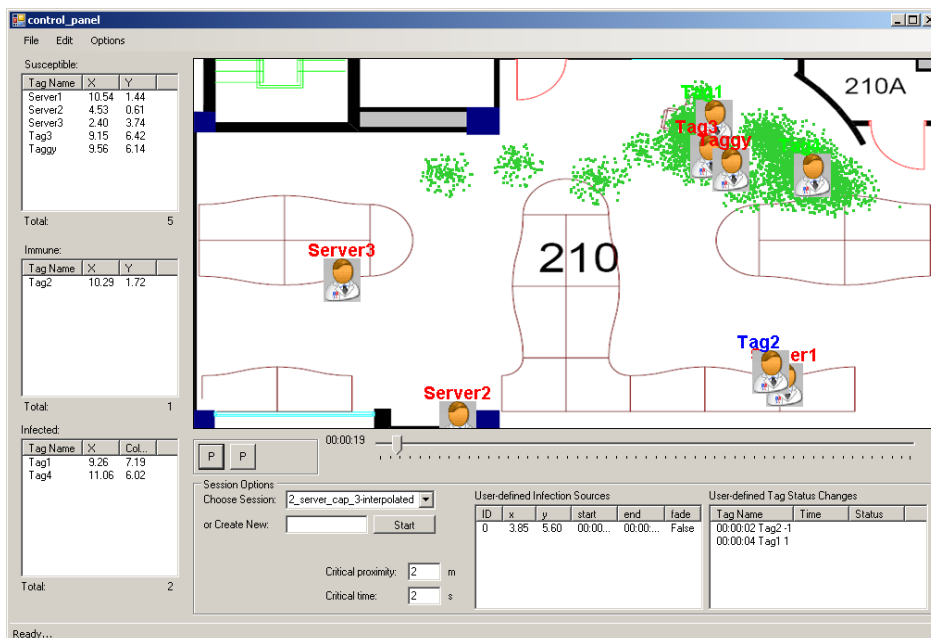


Figure 3.4: A later version of the application

The map sits at the top right hand corner of the GUI (figure 3.4). Each tag is displayed on the map in the correct position for the current time, as an icon depicting a person, with its name above. The text of the name is colour-coded according to its infection status: red for susceptible, green

for infectious and blue for recovered. An area in which a tag might become infected (for example the area around an infectious tag), is depicted by a cloud of green dots. The map is capable of displaying multiple tags and infection sources.

To the left of the map lie the tag status windows (for example the susceptible window shown in figure 3.5). Between them, these contain the names of all of the tags, and also display their x and y coordinates, which are constantly updated. From top to bottom, the three status windows correspond to susceptible, infectious and recovered tags. As a tag's status changes, it moves to the correct window, and the window's total (displayed just below each one) is updated accordingly. This allows the user to see at a glance both the location and infection status of any given tag.

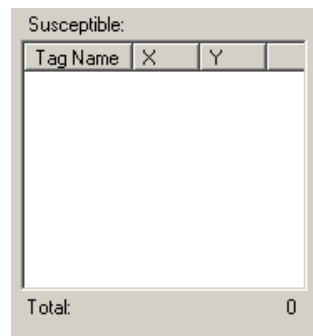


Figure 3.5: The susceptible tag status window

The time bar (figure 3.6) is displayed just below the map, and allows the user to move both forwards and backwards freely through the session. It is possible to move between different times simply by dragging the indicator along the bar. The current time is also displayed in text form just to the left of the bar, to give a more precise view.



Figure 3.6: The time bar, and play and record buttons

Below the time bar are the Infection Source window, which lists each infection source added by the user; and the Tag Status Change window, which lists each change in status of a tag that is actioned by the user (both shown in figure 3.7). Both of these functions are explained more fully in the next section.

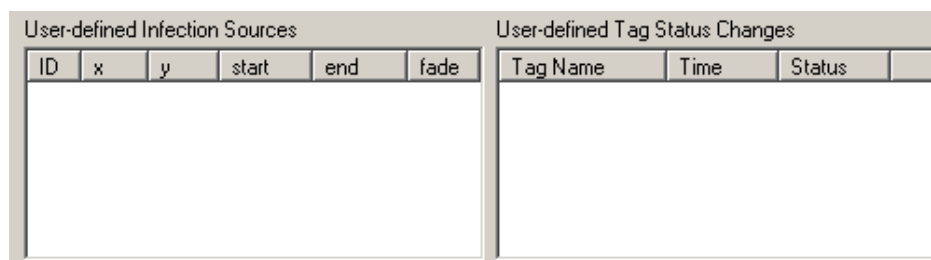


Figure 3.7: The Infection source box and tag status windows

Finally, there are various controls to start and stop playback and recording, and to create new, open and save sessions. These are located to the left of the time bar and in the *File* menu respectively.

My main focus while implementing the GUI was on ease of use. I aimed to create an interface that was intuitive, while still being fully-featured. This is reflected in features such as the ability to drag and drop tags from one status window to another, which are both powerful and easy to use.

### 3.3.3 Functionality

#### Record

Clicking on *File* → *New* notifies the program that the user wishes to record a new data set. By clicking on the red *record* button, the process is started, and the program starts waiting for data. Before a tag's data will be recorded, however, it must be activated. This is achieved by pressing a button on the tag twice. This adds the tag's serial number to the tag list, and new locations will now be recorded to the program in the following manner: Each time a new location is recorded by the ubisense system, it raises an event that is handled by an event handler in a separate process from the main program. The handler constructs a datatable row that contains the tag name, time, and x, y and z co-ordinates, and then passes this row asynchronously to the main thread. The asynchronous nature of the communication allows for simultaneous playback and record functionality, which will be discussed later. It also makes it possible to periodically back up the data to the database, to avoid all data being lost in the event of a crash.

Pressing the button on one of the tags being monitored invokes a separate event handler, which adds a row to a separate table. The button on the compact tags (and one of the buttons on the slim tag), are set by default to invoke a 'sneeze' action. The event handler adds a row to the infection sources table that includes the tag name and time of the sneeze (the position of the sneeze is derived from the tag's position at that time). The other button on the slim tags is set by default to invoke an 'immune' action (i.e. when the patient has been treated/innoculated). However, both of these functions can be changed quite simply. Any button press is acknowledged by a short beep the the program causes the tag to emit when it processes the button press event.

While a session is being recorded, it is played in real time on the map. This allows the user to monitor the session, and correct any problems that might occur. Sneezes are represented by a cloud of dots, in the same way as infections during playback mode (as in figure 3.8).

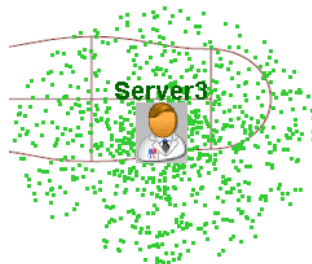


Figure 3.8: Green dots representing an infectious area centered on Server 3, whose name is coloured green to indicate that he is infectious

When the user has finished collecting data, he presses the record button for a second time, and the program stops recording new data. By selecting *File* → *Save*, he can then save this data to the database for later review.

## Play

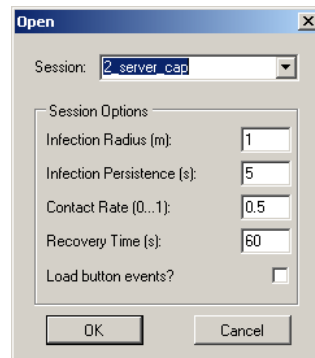


Figure 3.9: The *Open* Dialog Box

Clicking on *File* → *Open* brings up the *Open* dialog box (figure 3.9). This allows the user to load in any session that has been saved in the database. The dialog box also allows the user to define several parameters regarding the current session. These are:

- Infection Radius - How close two individuals have to be before there is a chance of passing an infection between them (they have to maintain this for two seconds, in order to exclude errors in the hardware and, for example, the act of simply passing in a corridor).
- Infection Persistence - How long the area that an infected person has been in remains infected after they have left it. The radius of a persistent infection decreases over this time until it is zero.
- Contact Rate - the likelihood that an infectious person will infect a susceptible person if they are within the infection radius for more than two seconds. This is different depending on the infection being studied.
- Recovery Time - the time before an individual's immune system deals with an infection and they become 'recovered'.
- Load button events checkbox - Whether to load events triggered by tags' buttons being pressed (e.g. sneezes).

On loading, the program creates a separate tag object for each tag listed in the session, and uses a dictionary to keep track of them by name. It also creates sneeze objects, which extend the functionality of more general infection source objects, and stores them in an arraylist. Each object contains methods such as `draw()`, which allow a good level of abstraction to be maintained in the main program.

Once a session has been loaded, the user is free to scroll through it by dragging the time bar control. He can also choose to play through the session. In order to change the infection status of a tag or tags at any particular time, he simply drags the time bar to that time, and then drags the tag name to the correct box. The infection status is changed immediately on the map. In addition, the details of the change are listed in the tag status changes box. This means that a status change can be removed later. This is accomplished by clicking on the status(es) to change, and pressing the delete button.

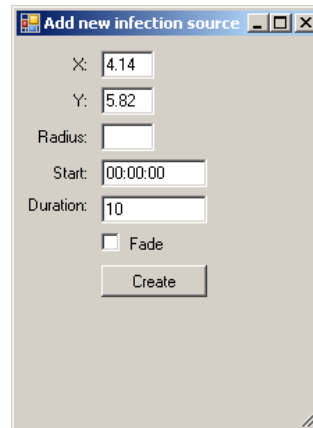


Figure 3.10: Infection Source Dialog Box

The user can also choose to add sources of infection, by clicking on the map where he wants the source to appear. This brings up a dialog box (figure 3.10), which allows him to define the following parameters:

- The X and Y co-ordinates (these are pre-filled with the values from the position on which he clicked, but can be altered here).
- The infection radius (i.e. how close a susceptible individual has to be to the infection source in order to have a chance of being infected).
- Start time - the time the source starts to be infectious
- Duration - how long the source is infectious for
- Fade checkbox - whether the radius of infection decreases throughout the duration, or remains constant.

As with the tag status changes, and infection sources that are added are displayed in the infection sources box. They can be deleted in the same way.

Whenever the time is advanced to a point beyond the time currently being shown on the map, the program runs through each database row in turn, and performs an algorithm (detailed in figure 3.11), to determine its status at that point. If the tag is deemed to be susceptible, and has been exposed to an infection source for long enough to constitute a contact, the program uses C#'s

pseudo-random number generator to generate a double between 0 and 1. If this is less than the user-defined contact rate, the tag status is set to infected.

Running through the data rows linearly ensures that the effect of being nearby infection sources and infected tags is properly included, and when the program has finished processing, the up to date and accurate statuses of each tag are shown. While this process does introduce a delay into the program, in most cases it is negligible. I considered several alternatives to this technique, the most promising being the creation of a list of contacts between tags and infection sources when the data is initially loaded from the database. However, this approach would have required separate lists to be kept for all different infection types, which would make it complicated to update the program should more be added later. Also, the addition or removal of an infection source would require a complete re-calculation of the contact list, which introduces a delay similar to that caused by the current algorithm. Another delay would be caused by the initial list creation when the data is loaded into the program.

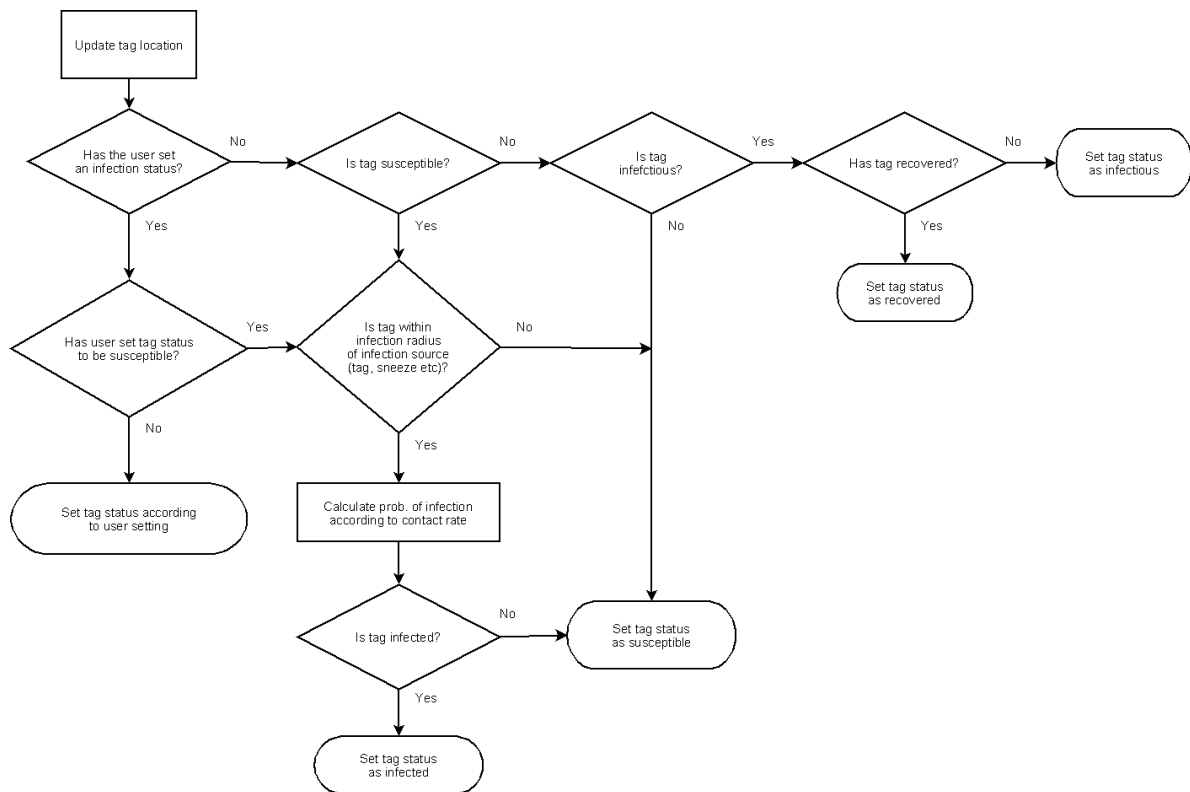


Figure 3.11: Algorithm to determine infection status

An added feature is that it is possible to jump to any previous time, and play the session from there. This is akin to a “pause live TV mode”, and means that analysis can be done on data even while new data are being captured. This is especially useful in cases where an extended data capture session is in progress, and results are needed quickly.

Once a user has defined the parameters to his satisfaction, and has advanced the time bar to the desired point, he is able to plot an SIR-style graph of the session so far (figure 3.12 for example). This allows him to compare sessions where different parameters have been set easily, and also allowed me to compare the software with the results from a standard infection model. As with

the map, the graph is plotted using Microsoft's GDI package. It is also totally self-contained in a separate class, and as such is easy to add to, should the model be extended to include more information in the future.

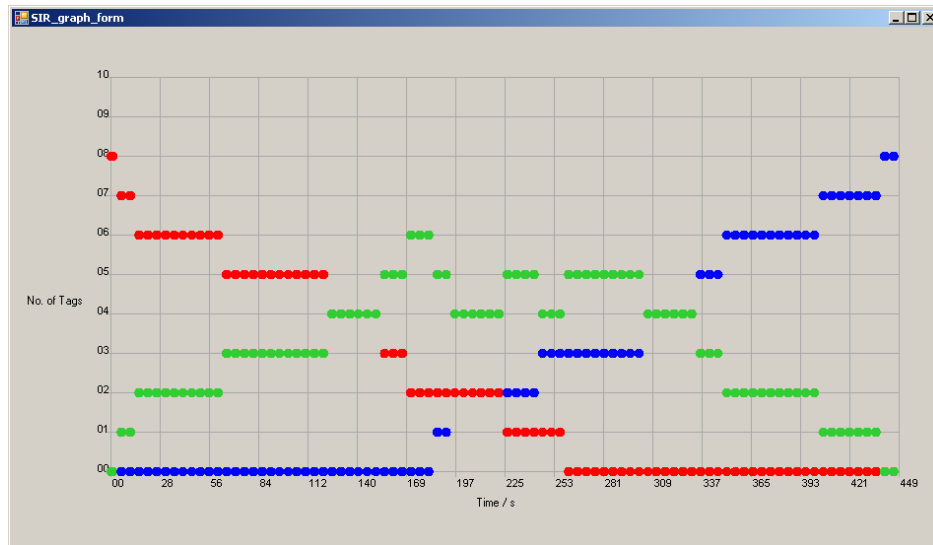


Figure 3.12: An SIR-style graph generated by the software. Red: susceptible, Green: infectious, Blue: Recovered





# Chapter 4

## Testing

### 4.1 The Ubisense System

In order to obtain a quantitative analysis of the Ubisense hardware's accuracy, I set up an experiment. I split the lab into eight equal sections (figure 4.1), and placed the same tag at the same height at the centre of each section.

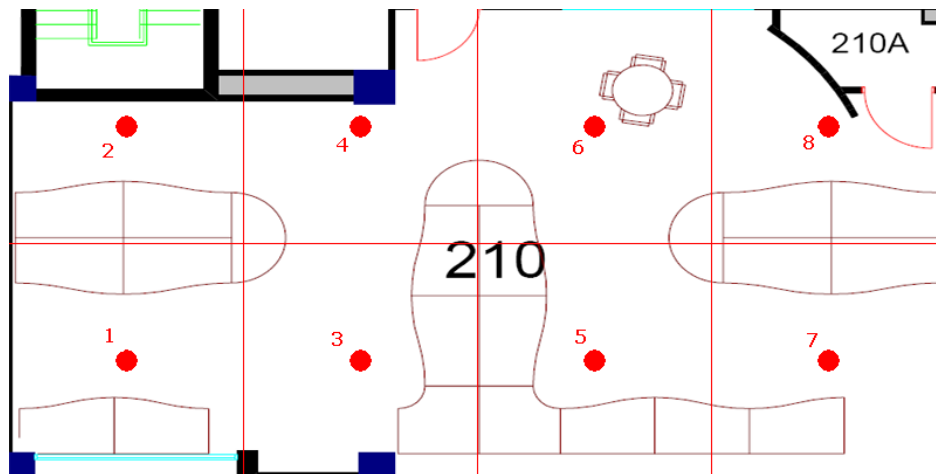


Figure 4.1: Accuracy analysis - red dots denote survey positions

I then recorded forty location measurements for each location, and averaged the forty to obtain a location reading. In order to eliminate systematic error due to incorrect calibration of the hardware, I performed a least-squares analysis on the difference between the observed and expected positions in both x- and y-planes, and added a correction factor for both that minimised the error. Finally, I calculated the difference in each corrected reading from the expected location, and averaged the errors. The full results are listed in the appendix, and the averaged readings are as follows:

Point	1	2	3	4	5	6	7	8
Error	0.077	0.608	0.111	0.184	0.138	0.253	0.331	0.287

The total average is therefore 0.25m, or 25cm, with an estimated experimental error of 3cm. I kept experimental error to a minimum by using a laser-measuring device.

## 4.2 The Software

In order to test that the software was accurately recording the locations of people in motion, I created a number of video recordings in which people carried tags around the room. I then compared these to the location data captured by the software, and made a visual evaluation of the accuracy. I found that while there were occasionally periods where locations were recorded inaccurately, on the whole the software accurately reflected the positions of the tags.

In order to test the effect of changing the four infection model variables (infection radius, infection persistence, contact rate and recovery time), I collected a sample data set. This was set up for a colleague's experiment, and involved creating a shop environment, with three members of staff. Four visitors entered the shop in turn, with arrival time following an *exponential*(1) distribution. Then were then served by Server 1 for a time also determined by an *exp*(1) distribution, before moving on to either Server 2 or Server 3 with probability 0.7 and 0.3 respectively, and waiting there for a number of minutes determined by *exp*(1.5) and *exp*(1.2) distributions. A typical situation during the experiment is depicted in figure 4.2, where Person C is being served by Server 1, and Person B is being served by Server 2.

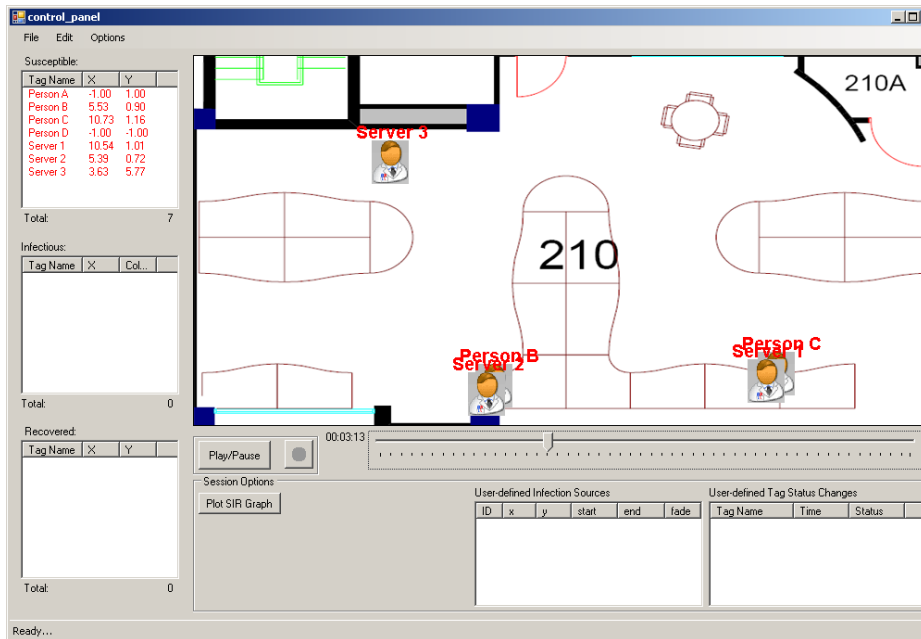


Figure 4.2: A screen capture showing locations from the sample data set

Using this data set, I ran a control experiment using the following parameters:

- Person A was infected at time 20 seconds
- The infection radius was 1 metre

- The infection persistence was 0 seconds
- The contact rate was 0.5
- The recovery time was 180 seconds

I then generated an SIR graph for this experiment (figure 4.3), which shows that in total five tags were infected, with all recovering within 390 seconds.

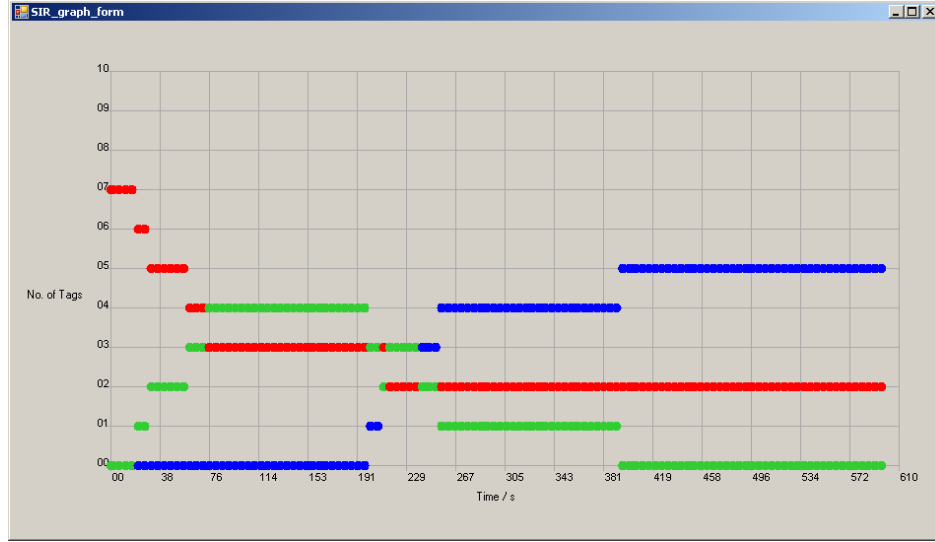


Figure 4.3: SIR graph for the control experiment (red: susceptible, green: infectious, blue: recovered)

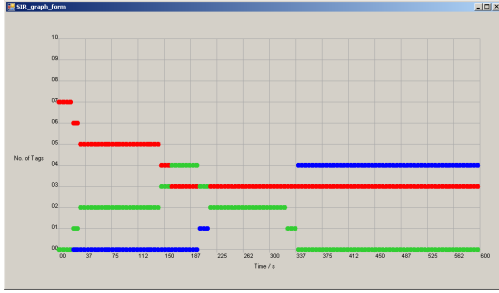
I then carried out four sets of experiments, in which I varied one parameter in each set, and obtained an SIR graph for each. Varying one parameter at a time ensured that any change was due to that parameter alone, and not to a combination of changes. However, some non-determinism was inevitably introduced by the probabilistic effect of the contact rate. The results of the experiments were as follows:

#### 4.2.1 Infection radius

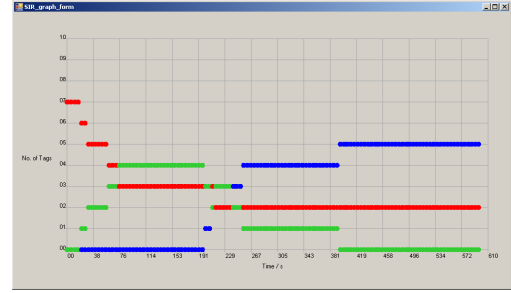
I ran simulations using values of 0.5, 1.5, 2, 2.5 and 3m for the infection radius (I did not carry out a simulation using 1m, as I had already used this value for the control). The SIR graphs generated from these are shown in figure 4.4.

From these graphs, it is obvious that as the infection radius increases, the overall number of tags infected increases. This is particularly marked between 1m (4.4(b)) and 1.5m (4.4(c)), where there is a jump from five to seven tags infected. This number remains constant from 1.5m (4.4(d)) to 3m (4.4(f)), although the peak of the infection (the point at which the number of infectious tags is highest) occurs progressively earlier.

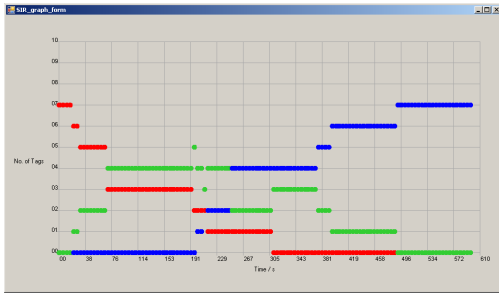
Although the infection radius is not a standard parameter that can be altered in the SIR equations,



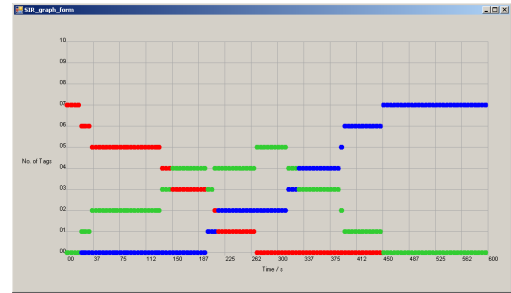
(a) Infection radius = 0.5m



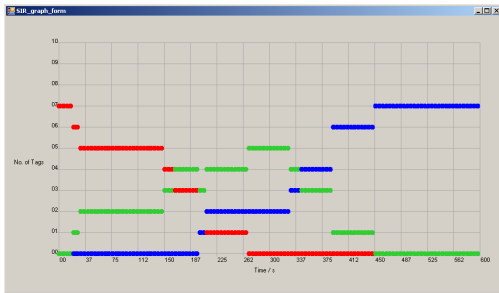
(b) Infection radius = 1m (Control experiment)



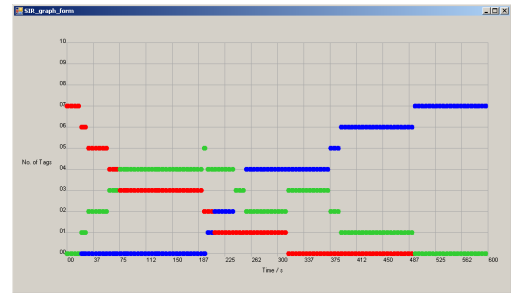
(c) Infection radius = 1.5m



(d) Infection radius = 2m



(e) Infection radius = 2.5m



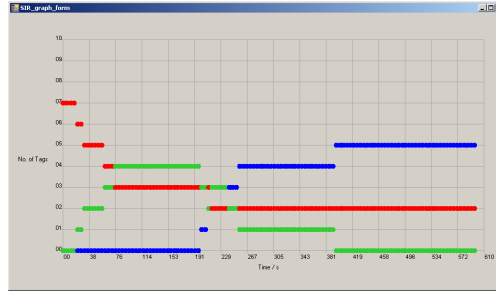
(f) Infection radius = 3m

Figure 4.4: SIR graphs for increasing infection radius. Red: susceptible, Green: infectious, Blue: recovered

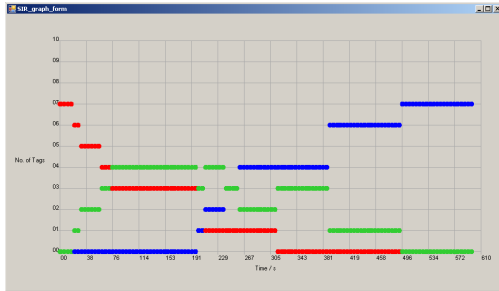
it is logical that as the infection radius increases, so the number of people infected will increase. In addition, it is likely that people will be infected sooner, as more fleeting contacts between individuals will become infection-spreading contacts.

#### 4.2.2 Infection Persistence

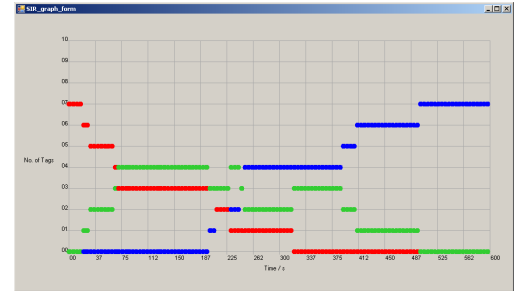
To test the effect of varying the infection persistence (the amount of time that an area remains infectious after an infectious person has moved through it), I ran simulations using values of 15, 30, 45 and 60 seconds for the infection persistence. The resulting SIR graphs are shown in figure 4.5.



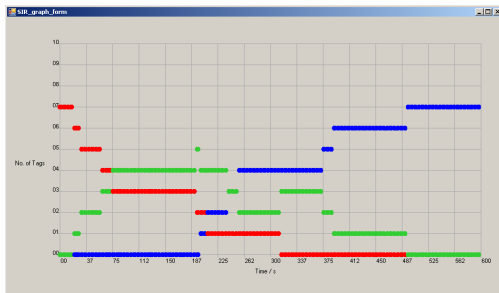
(a) Infection persistence = 0s (Control Experiment)



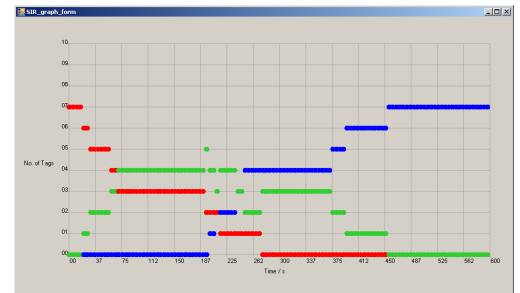
(b) Infection persistence = 15s



(c) Infection persistence = 30s



(d) Infection persistence = 45s



(e) Infection persistence = 60s

Figure 4.5: SIR graphs for increasing infection persistence. Red: susceptible, Green: infectious, Blue: recovered

As with the increase in infection radius, it is clear from the graphs that the total number of tags infected increases from five to seven when the infection persistence is increased from zero (comparing 4.5(a) to 4.5(b)). However, there is little difference when the infection persistence is increased above this point. This may be attributed to the fact that there is a relatively low number of individuals in my experiment, meaning that nobody crosses the path that an infected person has taken that he

would not otherwise come into contact with. This is impossible to evaluate without further study with a larger sample size.

### 4.2.3 Contact rate

The final two parameters (contact rate and recovery rate) are included in the standard SIR model. This means that the experiments where I varied these factors serve two purposes: as above, they can be used directly to observe the results of varying the parameters, but they can also be used to compare the effects against a standard SIR graph that has been generated using the system of differential equations stated in section 2.1.

I ran three simulations in which I increased the contact rate in increments of 0.25. The results are shown in figure 4.6, where they are compared to plots of the corresponding differential equation systems. I generated these plots using the statistical environment R<sup>1</sup> and an SIR modelling module [3].

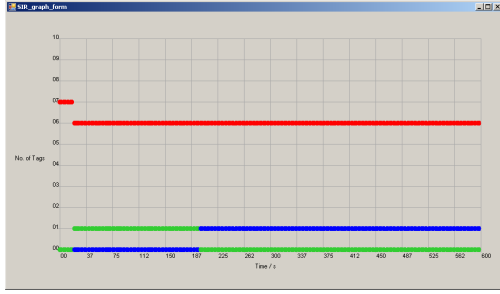
Looking at the graphs generated from the SIR equations, two effects are apparent when the contact rate is increased. The first is that the peak number of infectious individuals is higher. The second is that this peak is reached sooner the higher the contact rate. The first of these effects can be observed in the experiment data, although not as clearly as in the calculated graphs. Between figure 4.6(a) and figure 4.6(c), the peak number of infectious tags increases from one to five. However, it appears that there is little difference in the time that it takes for these peak values to be reached. This discrepancy from the calculated graphs is likely to be due to the necessarily small sample size, and discrete nature, of the experiment data. However, it could also be indicative of a problem with the design of the software. As with the infection persistence experiment, further study is needed to determine the cause.

### 4.2.4 Recovery Rate

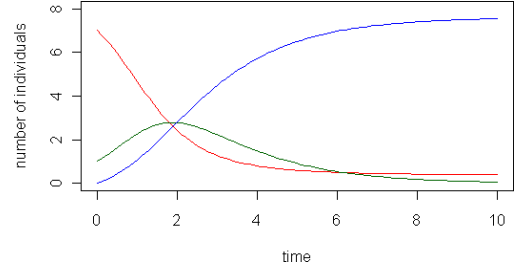
Due to the nature of the way the location data are recorded, the recovery rate set in the program does not correspond with that used in the actual SIR equations. The former is defined as a number of seconds, while the latter is expressed as a probability. The reason for this discrepancy is that the experimental data are discrete (i.e. they are recorded at discrete time intervals, the periodicity of which varies due to the Ubisense hardware), which makes it extremely difficult to calculate a continuous probability. However, broadly speaking, an increase in recovery time should correspond to a decrease in the standard SIR model recovery rate. Therefore, in order to compare this set of experiments to a calculated SIR graph, I estimated the recovery rate, and attempted to look for broad trends rather than a precise comparison. I ran simulations for recovery times of 60, 120, 180 and 240 seconds, the results of which can be seen in figure 4.7.

---

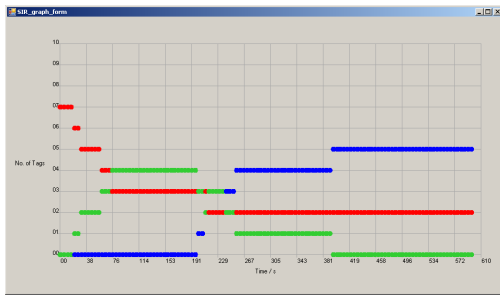
<sup>1</sup><http://www.r-project.org/>



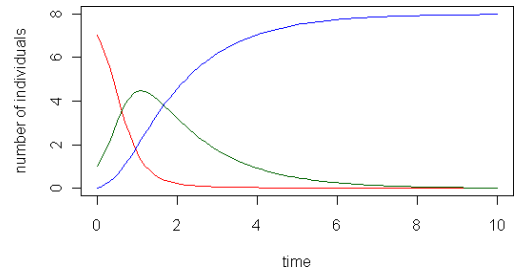
(a) Contact rate = 0.25 - Test data



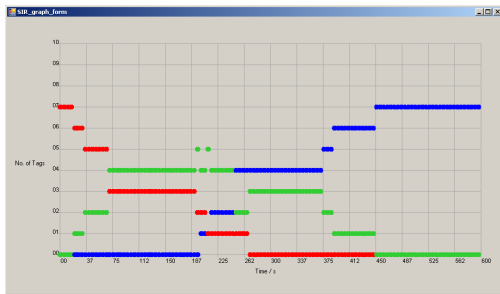
(b) Contact rate = 0.25 - Calculated values



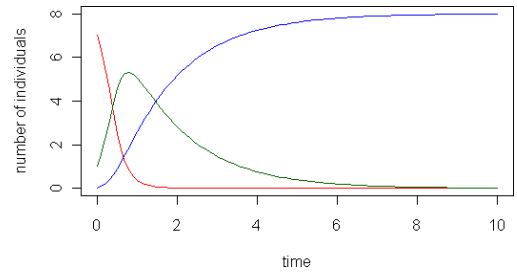
(c) Contact rate = 0.5 (control experiment) - Test Data



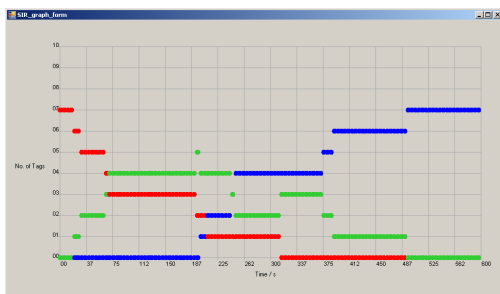
(d) Contact rate = 0.5 - Calculated values



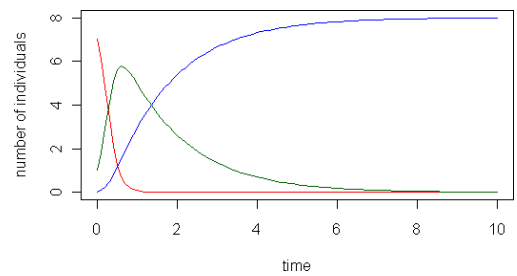
(e) Contact rate = 0.75 - Test Data



(f) Contact rate = 0.75 - Calculated values

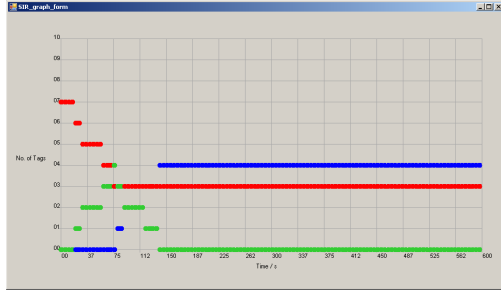


(g) Contact rate = 1.0 - Test Data

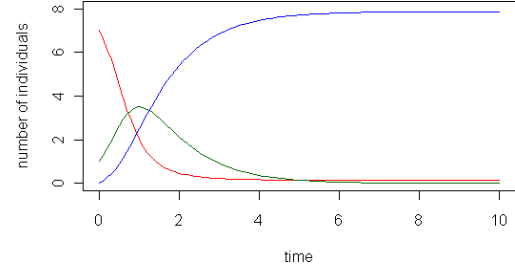


(h) Contact rate = 1.0 - Calculated values

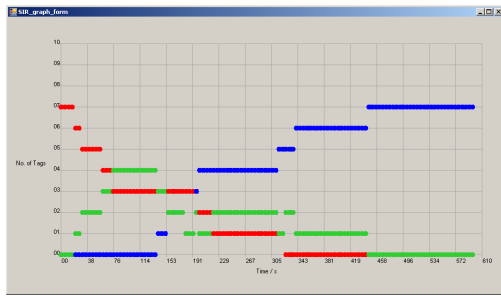
Figure 4.6: SIR graphs for increasing contact rate.



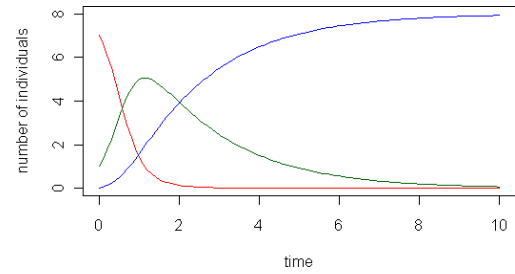
(a) Recovery time = 60 - Test data



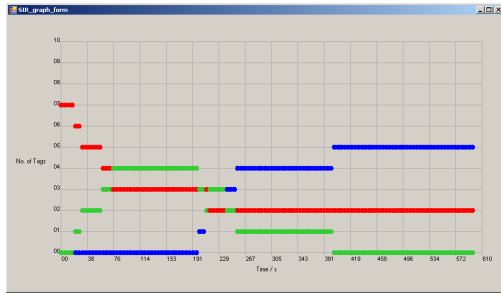
(b) Recovery rate = 1 - Calculated values



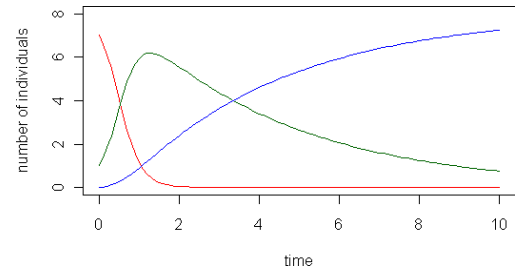
(c) Recovery time = 120 - Test data



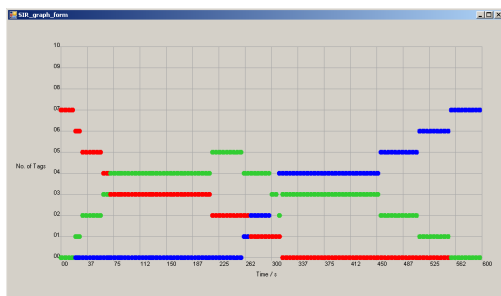
(d) Recovery rate = 0.5 - Calculated values



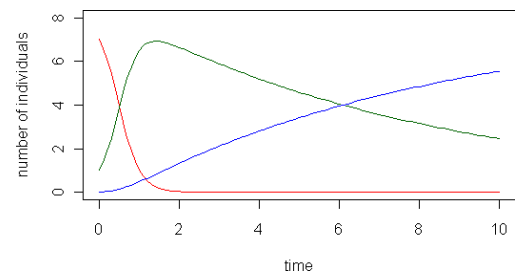
(e) Recovery time = 180 (control experiment) - Test data



(f) Recovery rate = 0.25 - Calculated values



(g) Recovery time = 240 - Test data



(h) Recovery rate = 0.125 - Calculated values

Figure 4.7: SIR graphs for increasing recovery time.



Two features of the calculated graphs are apparent. The first is that the gradient of the recovered line decreases as the recovery rate decreases (and therefore the recovery time increases). The second is that the peak number of infectious individuals increases as the recovery rate decreases, and after it has peaked, its gradient is less. Both of these trends can be seen in the experimental data, although the second more clearly than the first. The graphs show that as the recovery time increases, the initial gradient of the recovered line decreases. However, once tags have started to recover, the gradient increases, and is eventually very similar in all of the graphs. The peak number of infectious tags rises from four to six, and the gradient of the infectious line after its peak clearly decreases with increasing recovery time. The discrepancy between the gradient of the recovered line in the test data and calculated values could again be attributed to the small sample size, but again needs further study. However, the decrease in gradient of the infectious line shows a good correlation with the calculated data.



## Chapter 5

# Evaluation

### 5.1 The Ubisense System

#### 5.1.1 Setup

Setup of the Ubisense system is an involved process. First, enough sensors have to be installed (usually ceiling-mounted) to adequately cover the room. Each has to be connected to the main sensor of a cell by an ethernet cable, and individually to a network switch providing Power over Ethernet (PoE). When this has been accomplished, two lots of software have to be installed on the server, followed by installation of the Ubisense services that send and receive data. Following this, the sensors must be calibrated, which in itself involves measuring the precise location of each sensor, and measuring the background radio signal levels to avoid interference. Finally, the outline of a room can be drawn using the Site Manager package, and as much detail as is necessary can be added.

Once it is learned, the setup of the system is an easy, albeit lengthy, process. Fortunately, it should only need to be done once for every installation, barring the need to move sensors around.

#### 5.1.2 Operation

The Ubisense system claims to be capable of tracking tags with sub-second frequency to a sub-15cm resolution. During the time that I have been using the hardware, I have seen occasions when this is the case. However, the accuracy of reported locations does depend on several factors. These include:

- How well the tag is covered by the sensors

The fewer sensors that can 'see' the tag, the less accurate the position data will be. It is therefore important to place the sensors to provide maximum coverage of a room, and irregularly-shaped rooms, alcoves etc. can cause problems with this.

- The presence of UWB-reflective surfaces

Metal surfaces have a tendency to reflect UWB signals. While the Ubisense system employs software techniques to reduce inaccuracy caused by reflections, the occasional incorrect reading can be observed from the Location Engine Configuration screen.

- The presence of UWB-absorbers

UWB pulses are absorbed by water. The result of this is that tags cannot be tracked when obscured from the sensors by the human body. In our test setup, by making sure that tags are held away from the body, and that there are not many people in the room while experiments are carried out, it is easy to mitigate the effect of this. However, in a busier environment, when tags are carried in pockets or on belts, location data is likely to be incomplete at best, and quite patchy at worst. In a situation where it is essential to record all encounters between individuals, this is likely to cause problems with creating accurate analysis in a real-world situation, such as a hospital.

My own experiments revealed that the average accuracy of the Ubisense system was around 25cm, and having used the equipment for a number of weeks, I would say that this is a fair value. However, there are occasions when the readings jump rapidly between very accurate and widely inaccurate, which I have attributed to the hardware failing to filter out UWB reflections. Turning on the system's velocity filtering feature did help somewhat with this problem, but there was still the occasional erroneous signal.

I also had a number of problems receiving the data on a computer other than the server. Location and other data are transmitted by use of multicast packets over the network, which can then be received by other computers running the relevant Ubisense services. In practice, it took several tweaks to obtain a reliable data stream, including re-configuring IP addresses, disabling firewalls and adding static routes to the client's network routing table. As with the hardware setup, making the technology work to begin with can be challenging, but once accomplished should not need to be done again. It is worth noting that Ubisense run a training course to aid with the setting up of their hardware and software. Unfortunately I did not have a chance to attend this, but it would most likely aid the setup process.

## 5.2 The Software

In terms of receiving and storing accurate location data, the software is capable of receiving, processing and saving the location data for a limited number of tags for an arbitrary amount of time (in practice this is restricted by the amount of storage available). It is impossible to evaluate how the software would cope with data from a larger number of tags, but it is likely that the

capacity of the sensor network itself would be the limiting factor. A multi-cell network (in which several sensor networks communicate with the central server) may be more problematic, but without further testing it is impossible to tell.

In play mode, the software is capable of playing back the data it has recorded, along with button-press events such as sneezes, in real time. The location data is adequately represented on a map, which allows users to see each tag's location. If this system were expanded, it would be necessary to add features such as panning and zooming, but for the given scenario, the map proved perfectly adequate.

The software provides a comprehensive set of controls which allow the user to fully control the infection model. The combination of initial parameters (contact rate etc.), along with the ability to later define additional infection sources and change the infection status of tags, allows for maximum flexibility. These abilities, combined with the ease with which the user is able to scroll back and forth throughout the session, result in a feature-rich and intuitive application. Should more features be required later, it would be easy to extend the application to incorporate these due to the object-oriented design.

In order to evaluate the results of simulations, and compare them to existing models (such as the SIR model), the graph-plotting function provides an easily-readable analysis. However, given the small number of tags available for testing and the artificial nature of the environment, a realistic and fair comparison to existing models was difficult. Despite this, it is clear that the overall pattern of the models produced by the software are similar to those obtained from the standard SIR model. In addition, the limited number of trends that could be compared (those relating to the contact rate and to an extent the recovery time) indicate that altering these parameters in the software produces a similar effect to what would be expected.



## Chapter 6

# Conclusion

### 6.1 Overview

As a proof of concept, the combination of the Ubisense hardware and the software that I have designed is a success. It is clear that in a test environment at least, the hardware is capable of producing location readings that are sufficiently accurate to monitor the movement of individuals, to the extent that a contact tracing study that provides meaningful results can be performed. Given the small-scale nature of the test environment, the fact that any results that mimic real-life infections can be obtained at all is very promising, and it is likely that a larger-scale experiment would improve on these results. The software achieves its objectives to provide an easy to use and fully-featured interface to explore the effect of different parameters on the spread of an infection throughout a given environment.

Given the relative youth of the technology currently used, it is also likely that the accuracy of location tracking information will increase as more progress is made in the field. This can only help to improve the reliability of such models in the future. Likewise, as the concept of using location tracking data as a basis for contact tracing is relatively new, and I have designed the algorithms that determine the infection status of individuals from scratch, these can undoubtedly be improved and expanded upon, with the result that they will more closely match the known behaviour of infections. It is my hope that such a program could even help epidemiologists to learn more about the nature of infections, and how they spread.

### 6.2 Obstacles

The main obstacle I encountered during my project was the setup and calibration of the Ubisense hardware. The hardware itself required careful calibration, which in itself was a time-consuming activity. Once this had been accomplished, I was required to apply a largely trial and improvement

technique to set up the Ubisense software in such a way that data was transmitted over the network. Despite my efforts, one of the sensors crashed on a regular basis and required re-booting. In addition, from time to time the system would stop working, and would require re-calibration.

Once the hardware was set up, I then spent a large amount of time familiarising myself with the Ubisense C# API. This was an especially steep learning curve, as I was learning C# itself at the same time. The documentation provided for the API is extremely limited, and eventually I obtained most of the information that I required from programming examples available online at the Ubisense Research Network forum<sup>1</sup>.

In terms of designing the software, the most challenging obstacle to overcome was the sheer volume of information that needed to be stored and processed in order to record and play back a set of location tracking data. The unpredictable nature of contacts between individuals meant that in order to provide maximum flexibility to the user in defining parameters and then being able to jump instantly to any point in the session, I had to devise an algorithm that could run through a given portion of a recorded session at high speed, calculating tag infection statuses on the fly. In many cases, this means applying the algorithm shown in figure 3.11 to hundreds of thousands of location points. However, I believe I have devised an efficient system which works reliably and accurately, a view which is borne out by the reliable nature of the software. In addition, the fact that the bulk of the calculation is carried out in one main algorithm means that it is easy to add further elements to this should it be necessary.

## 6.3 Future Work

Given the innovative nature of this project, there is undoubtedly a large amount of further work to be done regarding the use of location tracking in infection modelling. My software is largely a proof of concept, and requires much more extensive testing than I have been able to perform in the lab with a limited number of tags. Ideally, this would involve covering a hospital with an Ubisense multi-cell network, and capturing data over a long period. This could then be used as a basis for improving the software and the algorithms used in it. However, the logistics and cost of doing this at present are considerable, and it may be some time before the hardware is mature enough to report accurately the amount of data that would be required for a large-scale study.

In particular, it should be possible to extend the SIR model that I used to include more variables. The reason I started with the SIR model was its simplicity, which meant that I was able to write a working program and obtain results in a short space of time. In reality, a person suffering an infection goes through more states, and there are more factors affecting the the infection's spread. The SEIR model, for example, includes an incubation period, where an individual has been infected but is not yet infectious. It should also be possible to include births and deaths, which would of course lead to a variable population size. These extra variables should be easy to add to the program, but each requires testing to ensure that it is implemented correctly.

---

<sup>1</sup><http://www.ubisense.org>



One of the objectives stated in the aims section of this report that I would have liked to have pursued further is that of producing an ordered list of the individuals in a given environment according to how many contacts they have with others. This could provide invaluable information for doctors, allowing them to target immunisation and treatment at those who interact with a large number of people. It should also be possible to produce a contact graph, such as figure 2.2, listing every contact between every monitored individual in a certain period of time. Unfortunately I was not able to include this functionality due to time constraints, but it should not be difficult to extend the software to include this.

Aside from the obvious applications in the healthcare sector, there are undoubtedly other uses to which software of this kind could be put. It essentially tracks social interactions, and as such could well have implications in the study of psychology. For example, the exchange of information between people can be analogised to the spread of an infection. As any technology matures, more and more applications become apparent, and there is little reason to think that location tracking will be any different.



# Bibliography

- [1] Demonstrations of ubisense solutions in action. Website: [http://www.ubisense.net/index.php?load=content&page\\_id=80](http://www.ubisense.net/index.php?load=content&page_id=80). Retrieved on 14/07/2008.
- [2] Ekahau - wi-fi based real-time tracking and site survey solutions. Website: <http://www.ekahau.com/?id=1010>. Retrieved on 14/07/2008.
- [3] Florence Debarre. Sir models of epidemics. Website: <http://www.tb.ethz.ch/education/model/SIR>. Retrieved on 10/09/2008.
- [4] Ramon Huerta and Lev S. Tsimring. Contact tracing and epidemics control in social networks. *Physical Review E*, 66(5):056115.1–056115.4, 2002.
- [5] M. Kretzschmar J. Muller and K. Dietz. Contact tracing in stochastic and deterministic epidemic models. *Math. Biosci.*, 164:39–64, 2000.
- [6] S. Choi K. Yun and D. Kim. A robust location tracking using ubiquitous rfid wireless network. pages 113–124, 2006.
- [7] W.O Kermack and A.G McKendrick. A contribution to the mathematical theory of epidemics. 1927.
- [8] F. Javier Nieto Moyses Szklo. *Epidemiology: Beyond the Basics*, chapter 1, page 3. Jones & Bartlett, 2007.
- [9] S. R. Strom. Charting a course toward global navigation. The Aerospace Corporation: <http://www.aero.org/publications/crosslink/summer2002/01.html>. Retrieved on 02/09/2008.



# Appendix A

## User Guide

### A.1 Introduction

This software is designed to allow the collection and analysis of location tracking data with a view to creating infection models. It combines the techniques of SIR modelling and Contact Tracing to predict the spread of an infection amongst a population, given certain parameters about the infection that can be set by the user.

The software uses the Ubisense Location Tracking System to record the locations of any number of pager-sized ubitags within an Ubisense sensor network. These ubitags can be carried easily by individuals, and thus their locations can be monitored. This data is then saved in a database, and can be played back as necessary, and overlaid with a wide range of variables relating to an infection. The intention is that by creating various scenarios, the user can predict what would happen if an epidemic were to occur in a given environment.

### A.2 Requirements

The minimum hardware and software requirements for the software to run are:

- Windows XP or later, with the .NET 2.0 Framework installed
- An SQL-compliant server, preferably Microsoft SQL Server
- For the record functionality, at least one Ubisense location cell with accompanying client services running. Note that the software will work without this, but you will not be able to record location data.

## A.3 Getting Started

The software consists of one main window, the control panel. All aspects of the software can be controlled from here. The control panel is split into four main parts:

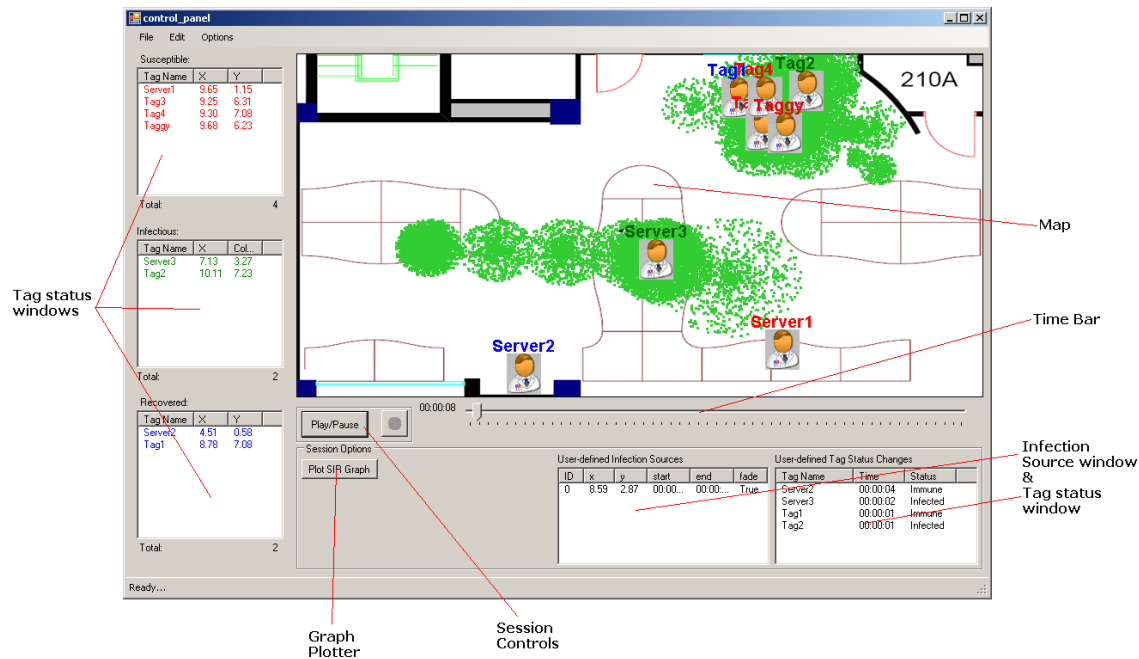


Figure A.1: A screenshot of the Control Panel, showing the different sections

- The map window. This is the main focus of the control panel, and allows you to view the location and infection status of tracked tags in both record and playback modes.
- The tag status windows. These contain a list of all the active tags, and denote their current infection status. There are three windows: one listing susceptible tags (those who are at risk of being infected), one listing infectious tags (those who are infected and can infect others), and one listing recovered tags (those who have recovered from the infection, and are no longer susceptible or infectious). These are colour-coded red, green and blue respectively.
- The infection source and tag status change windows. In playback mode, these list all of the user-defined infection sources and instances where the user has changed the infection status of a tag.
- The time bar and session controls. These allow the user to move back and forth through a session at will, and to start and stop recording or playback.

Additionally, there is a button that generates an graph of the current session, showing the number of tags of each status against time. There are also further controls in the *File* menu at the top of the control panel, which will be discussed further in later sections.

## A.4 Modes of Operation

The software has two modes of operation: record and play. The first allows location data to be recorded from an Ubisense sensor network, while the second allows such data to be viewed on the screen.

### A.4.1 Record

To start a record session, select the *File* menu and click *New*. There will be a slight pause while the software connects to the Ubisense server. Please note that in order for this functionality to be used, the computer must have the Ubisense client services running. For further information on this, please refer to the Ubisense manual.

When the software has connected, the *Record* button will be enabled and will turn red. To start recording, press the *Record* button. Once the button has been pressed, each tag to be tracked must be registered with the session. This can be achieved by pressing any button on the tag twice. This will cause the tag's serial number to be displayed in the susceptible status window, and its position to be displayed on the map.

As each tag is moved around the sensor network, its position will be updated on the map. The tag buttons are enabled, and can be used to mark events. The single button on the compact tag, and the lower button on the slim tag, simulate a sneeze. This is displayed on the map as a cloud of red dots, and the event is saved in the program. The upper button on the slim tag simulates treatment or immunisation, and immediately changes the tag's status to recovered. This change is reflected in the tag status windows and map.

At any point during recording, pressing the *Play/Pause* button will return the time bar pointer to the start of the session and the tag locations and events will start playing on the map. The behaviour of the software then mimics that of playback mode, which is discussed further below. At the same time, recording is still carried out in the background, and tag locations are captured as normal. Pressing the *Play/Pause* button again will return the software to standard record mode.

When enough data has been recorded, pressing the record button for a second time will stop the recording process. The data can then be saved to the database by clicking on the *File* menu, and then on *Save*. A dialog box will prompt you for a session name so that you can find the data later.

### A.4.2 Playback

Playback mode allows you to view data that has been previously recorded by the system. Clicking on the *File* menu and then on *Open* brings up the Open dialog box (figure A.2).

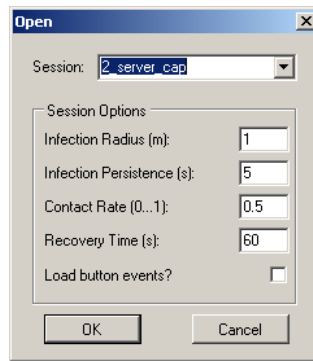


Figure A.2: The *Open* Dialog Box

This allows you to select which saved session you wish to play back, and allows you to set the variables relating to the infection for the session you are about to play. These variables are:

- Infection Radius - How close two tags have to be before there is a chance of passing an infection between them (they have to maintain this for two seconds, in order to exclude errors in the hardware and, for example, simply passing in a corridor).
- Infection Persistence - How long the area that an infected tag has been in remains infected after it has left it. The radius of a persistent infection decreases over this time until it is zero.
- Contact Rate - the likelihood that an infectious tag will infect a susceptible tag if it is within the infection radius for more than two seconds. This is different depending on the infection being studied.
- Recovery Time - the time before an infectious tag's status is changed to *Recovered*.
- Load button events checkbox - Whether to load events triggered by tags' buttons being pressed (e.g. sneezes).

Once the variables have been set, click on *OK*. There will be a slight pause as the location data for the selected session is loaded, a process which can be monitored by the status bar at the bottom of the Control Panel. Once the data has been loaded, the *Play/Pause* button will become active.

Clicking on the *Play/Pause* button will start the session from the beginning. It will play in real time, displaying the locations of each tag in the session on the map window, and its status in the status windows. Pressing the *Play/Pause* button again pauses the session, and the time, locations and statuses will be frozen.

The time bar displays how far through the session the current point is. Dragging the pointer to a different point on the bar will cause the program to jump to that point in the session, and the tag statuses and locations will be updated to reflect this. When jumping to a point after the current time, there may be a slight delay while the program calculates the new tag statuses.



Each tag starts off as susceptible. As the session is played back, a tag's status may change because of any of the following factors:

- The user changes the tag's status
- The tag has been infected by another tag or an infection source
- The tag's status was changed to recovered by a button press during record mode
- The tag has been infectious for the duration of the recovery time, and its status has changed to recovered.

Any change in a tag's status is reflected immediately both on the map and in the tag status windows.

The infection status of a tag may be changed at any time by clicking on the appropriate tag's name in the status window, and dragging it to the desired status window. This action takes effect immediately, and the tag's name on the map will change colour according to its new status. In addition, this change will be entered into the tag status change window, which displays the tag name, the time of the status change, and the status. Any change can be deleted by highlighting it and pressing the *delete* key on the keyboard. This will retroactively change the tag's status, and re-calculate the infection status of each tag given the change. This process may introduce a slight delay if deleting a status change that occurred a long time before the current session time.

A separate infection source may be added by clicking on the desired point on the map at any time. This will bring up the infection source dialog box (figure A.3), where you can define the following parameters:

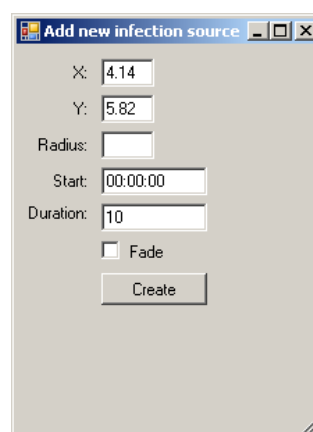


Figure A.3: The *Infection Source* Dialog Box

- The x and y coordinates of the centre of the source (defaults to the point that was clicked on)
- The infection radius. This is the radius from the centre within which tags may be infected.

- The time the infection source begins (defaults to the current time)
- The duration the source is infectious for.
- Whether the infection radius decreases over time.

On clicking *OK*, an entry will be added to the infection source window, which lists its properties and can be deleted in the same way as a tag infection status change.

At any point during playback, clicking on the *Plot SIR Graph* button performs an analysis of the tag infection statuses, and brings up a graph window A.4. This displays a plot of the number of tags of each infection status against time. This is colour coded in the same way as the tag statuses, i.e. red for susceptible, green for infectious, and blue for recovered. This graph can then be compared to other SIR plots.

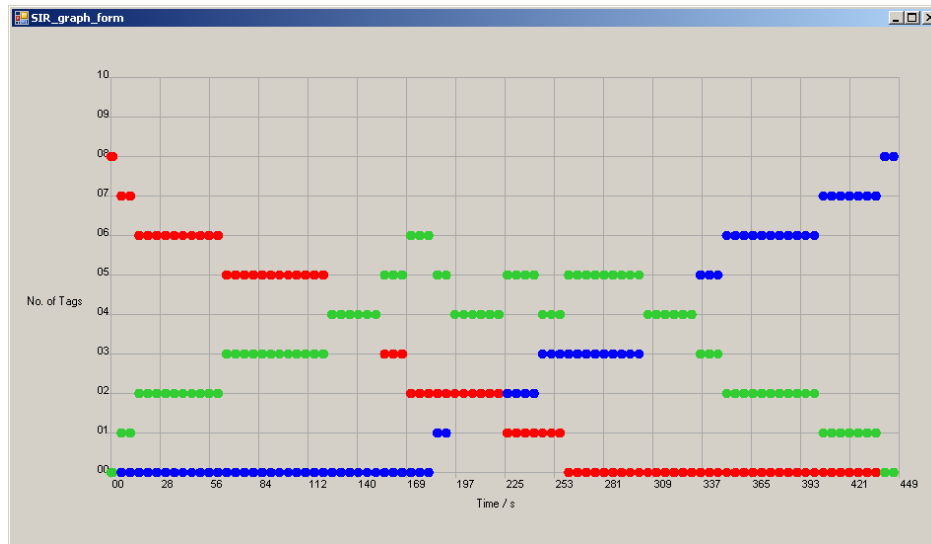


Figure A.4: The graph window

When you have finished playback of a session, you can save it by clicking on the *File* menu, and then clicking on *Save As...* The program can then be closed either by clicking on *File* and then *Exit*, or by clicking on the *x* at the top right hand corner of the control panel.

## Appendix B

### Class Diagram



# Appendix C

## Ubisense accuracy data

Point:	1		2		3		4		5		6		7		8	
Values:	2.180	1.885	3.151	5.459	5.614	1.667	5.696	5.972	8.894	2.083	8.800	5.892	12.171	1.787	12.349	5.760
	2.180	1.885	3.102	5.461	5.606	1.663	5.696	5.965	8.892	2.087	8.802	5.877	12.174	1.783	12.573	6.167
	2.171	1.912	3.062	5.459	5.598	1.660	5.699	5.959	8.905	2.082	8.797	5.893	12.169	1.810	12.556	6.155
	2.168	1.935	3.026	5.464	5.596	1.663	5.700	5.954	8.914	2.117	8.797	5.901	12.152	1.787	12.325	5.959
	2.159	1.953	3.007	5.456	5.592	1.659	5.712	5.948	8.900	2.131	8.796	5.892	12.160	1.783	12.325	5.958
	2.136	1.997	2.994	5.434	5.587	1.648	5.712	5.945	8.902	2.125	8.797	5.889	12.162	1.781	12.519	6.411
	2.116	2.041	2.959	5.460	5.584	1.694	5.706	5.953	8.897	2.139	8.795	5.906	12.163	1.781	12.325	5.884
	2.128	2.034	2.376	6.386	5.648	2.038	5.701	5.965	8.897	2.131	8.793	5.912	12.168	1.774	12.318	5.850
	2.130	2.040	2.376	6.386	5.722	2.154	5.704	5.952	8.886	2.122	8.796	5.908	12.173	1.776	12.326	5.846
	2.128	2.042	2.377	6.384	5.757	2.189	5.700	5.950	8.887	2.121	8.793	5.917	12.179	1.777	12.332	5.855
	2.125	2.060	2.414	6.324	5.745	2.176	5.702	5.946	8.880	2.111	8.789	5.925	12.181	1.775	12.336	5.833
	2.125	2.067	2.691	5.842	5.745	2.172	5.696	5.952	8.876	2.107	8.787	5.937	12.185	1.773	12.438	6.193
	2.127	2.070	2.654	5.871	5.734	2.162	5.697	5.953	8.869	2.102	8.792	5.916	12.183	1.773	12.719	6.451
	2.144	2.042	2.650	5.876	5.736	2.159	5.696	5.958	8.866	2.097	8.794	5.901	12.188	1.777	12.700	6.443
	2.109	2.103	2.658	5.837	5.725	2.161	5.697	5.953	8.871	2.094	8.798	5.889	12.187	1.773	12.403	6.126
	2.101	2.126	2.668	5.789	5.727	2.158	5.693	5.951	8.865	2.090	8.797	5.902	12.193	1.773	12.387	6.103
	2.109	2.101	2.668	5.772	5.716	2.162	5.692	5.947	8.859	2.082	8.792	5.918	12.191	1.777	12.273	5.587
	2.108	2.101	2.673	5.750	5.697	2.159	5.692	5.943	8.865	2.084	8.791	5.930	12.196	1.778	11.797	4.618
	2.111	2.099	2.681	5.722	5.700	2.159	5.697	5.929	8.859	2.086	8.792	5.923	12.196	1.780	12.802	5.047
	2.116	2.103	2.336	6.216	5.705	2.162	5.703	5.926	8.857	2.104	8.790	5.934	12.198	1.779	12.854	5.034
	2.119	2.126	2.376	6.147	5.707	2.168	5.701	5.924	8.860	2.100	8.790	5.943	12.199	1.778	12.310	5.780
	2.136	2.109	2.382	6.127	5.713	2.169	5.718	5.903	8.870	2.117	8.793	5.918	12.203	1.776	12.105	5.712
	2.130	2.127	2.685	5.519	5.697	2.159	5.721	5.893	8.864	2.111	8.794	5.903	12.204	1.775	12.041	5.904
	2.131	2.125	2.667	5.542	5.699	2.150	5.719	5.902	8.852	2.105	8.798	5.898	12.202	1.776	12.054	5.896
	2.138	2.104	2.649	5.572	5.692	2.143	5.713	5.913	8.858	2.094	8.802	5.887	12.200	1.776	12.111	5.806
	2.143	2.099	2.659	5.541	5.697	2.147	5.713	5.910	8.853	2.090	8.804	5.872	12.200	1.775	12.133	5.833
	2.136	2.132	2.653	5.562	5.701	2.154	5.711	5.904	8.848	2.086	8.801	5.894	12.195	1.775	12.106	5.770
	2.134	2.160	2.665	5.528	5.705	2.151	5.712	5.904	8.835	2.078	8.803	5.889	12.197	1.769	12.144	5.816
	2.130	2.181	2.680	5.495	5.698	2.151	5.721	5.902	8.834	2.076	8.803	5.891	12.199	1.772	12.155	5.823
	2.124	2.204	2.666	5.529	5.702	2.147	5.728	5.909	8.839	2.075	8.800	5.904	12.199	1.773	12.496	6.319
	2.132	2.190	2.641	5.570	5.698	2.144	5.725	5.909	8.836	2.073	8.800	5.892	12.203	1.767	12.560	6.575
	2.138	2.182	2.628	5.610	5.701	2.144	5.724	5.917	8.843	2.074	8.802	5.886	12.206	1.766	12.291	6.090
	2.137	2.175	2.635	5.619	5.707	2.143	5.720	5.923	8.840	2.068	8.800	5.897	12.207	1.761	12.292	6.097
	2.137	2.169	2.624	5.656	5.712	2.141	5.718	5.929	8.837	2.067	8.801	5.894	12.216	1.788	12.292	6.063
	2.135	2.192	2.627	5.639	5.717	2.140	5.716	5.925	8.843	2.063	8.796	5.901	12.213	1.788	11.949	4.963
	2.125	2.215	2.629	5.627	5.721	2.140	5.723	5.922	8.853	2.082	8.796	5.903	12.214	1.788	11.607	4.348
	2.125	2.216	2.642	5.604	5.724	2.139	5.717	5.928	8.849	2.079	8.796	5.906	12.215	1.783	11.318	3.395
	2.128	2.203	2.652	5.583	5.726	2.139	5.717	5.925	8.845	2.081	8.796	5.899	12.215	1.780	10.709	2.381
	2.150	2.150	2.664	5.563	5.721	2.135	5.713	5.930	8.842	2.078	8.799	5.895	12.216	1.779	12.207	5.827
	2.155	2.124	2.663	5.565	5.715	2.133	5.725	5.919	8.830	2.074	8.799	5.887	12.217	1.774	12.353	5.978
Observed:	2.134	2.095	2.675	5.724	5.692	2.065	5.709	5.933	8.864	2.094	8.797	5.903	12.192	1.778	12.247	5.691
corrected:	1.657	2.084	2.199	5.713	5.216	2.054	5.232	5.922	8.388	2.084	8.320	5.892	11.716	1.767	11.771	5.821
Expected:	1.700	2.020	1.700	6.060	5.110	2.020	5.110	6.060	8.510	2.020	8.510	6.060	11.930	2.020	11.930	6.06
corr. O - E:	-0.043	0.064	0.499	-0.347	0.106	0.034	0.122	-0.138	-0.122	0.064	-0.190	-0.168	-0.214	-0.253	-0.159	-0.239
Distance	0.077		0.608		0.111		0.184		0.138		0.253		0.331		0.287	
Average:	0.249															



# Appendix D

## Sotware testing data

### Experiment control

Infection Radius: 1

Persistence: 0

Contact Rate: 0.5

Recovery time: 180

21 - infected - Person A

27 - infected - Server 1

60 - infected - Person B

72 - infected - Person C

201 - recovered - Person A

208 - recovered - Server 1

210 - infected - Server 2

221 - infected - Server 3

240 - recovered - Person B

252 - recovered - Person C

271 - infected - Person D

## D.1 Infection radius

**exp 1.1 - infection radius = 0.5**

21 - infected - Person A  
31 - infected - Server 1  
142 - infected - Person C  
160 - infected - Person B  
201 - recovered - Person A  
212 - recovered - Server 1  
323 - recovered - Person C  
341 - recovered - Person B

**exp 1.2 - infection radius = 1.5**

21 - infected - Person A  
30 - infected - Server 1  
67 - infected - Person B  
68 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
211 - recovered - Server 1  
213 - infected - Server 3  
247 - recovered - Person B  
249 - recovered - Person C  
311 - infected - Person D  
373 - recovered - Server 2  
394 - recovered - Server 3  
491 - recovered - Person D

**exp 1.3 - infection radius = 2**

21 - infected - Person A  
34 - infected - Server 1  
132 - infected - Person C  
146 - infected - Person B  
201 - recovered - Person A  
209 - infected - Server 3



214 - recovered - Server 1  
214 - infected - Server 2  
267 - infected - Person D  
313 - recovered - Person C  
326 - recovered - Person B  
390 - recovered - Server 3  
395 - recovered - Server 2  
447 - recovered - Person D

**exp 1.4 - infection radius = 2.5**

21 - infected - Person A  
30 - infected - Server 1  
146 - infected - Person B  
162 - infected - Person C  
201 - recovered - Person A  
208 - infected - Server 2  
209 - infected - Server 3  
211 - recovered - Server 1  
269 - infected - Person D  
326 - recovered - Person B  
342 - recovered - Person C  
388 - recovered - Server 2  
389 - recovered - Server 3  
450 - recovered - Person D

**exp 1.5 - infection radius = 3**

21 - infected - Person A  
26 - infected - Server 1  
60 - infected - Person B  
74 - infected - Person C  
195 - infected - Server 2  
201 - recovered - Person A  
206 - recovered - Server 1  
208 - infected - Server 3  
240 - recovered - Person B  
254 - recovered - Person C  
312 - infected - Person D

375 - recovered - Server 2  
389 - recovered - Server 3  
493 - recovered - Person D

## D.2 Infection persistence

**exp 2.1 - persistence = 5**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
68 - infected - Person C  
201 - recovered - Person A  
208 - recovered - Server 1  
210 - infected - Server 2  
214 - infected - Server 3  
240 - recovered - Person B  
249 - recovered - Person C  
311 - infected - Person D  
390 - recovered - Server 2  
394 - recovered - Server 3  
492 - recovered - Person D

**exp 2.2 - persistence = 10**

21 - infected - Person A  
31 - infected - Server 1  
64 - infected - Person B  
76 - infected - Person C  
201 - recovered - Person A  
211 - recovered - Server 1  
221 - infected - Server 3  
244 - recovered - Person B  
256 - recovered - Person C  
267 - infected - Person D  
315 - infected - Server 2  
401 - recovered - Server 3  
448 - recovered - Person D

495 - recovered - Server 2

**exp 2.3 - persistence = 15**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
76 - infected - Person C  
201 - recovered - Person A  
206 - infected - Server 2  
208 - recovered - Server 1  
210 - infected - Server 3  
240 - recovered - Person B  
256 - recovered - Person C  
311 - infected - Person D  
387 - recovered - Server 2  
391 - recovered - Server 3  
492 - recovered - Person D

**exp 2.4 - persistence = 20**

21 - infected - Person A  
31 - infected - Server 1  
60 - infected - Person B  
72 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
210 - infected - Person B  
210 - infected - Server 1  
210 - infected - Server 3  
211 - infected - Person B  
213 - infected - Person C  
267 - infected - Person D  
373 - recovered - Server 2  
390 - recovered - Server 1  
391 - recovered - Server 3  
391 - recovered - Person B  
393 - recovered - Person C  
448 - recovered - Person D

**exp 2.5 - persistence = 30**

21 - infected - Person A  
31 - infected - Server 1  
64 - infected - Person B  
68 - infected - Person C  
201 - recovered - Person A  
210 - infected - Server 3  
211 - recovered - Server 1  
227 - infected - Server 2  
244 - recovered - Person B  
249 - recovered - Person C  
318 - infected - Person D  
391 - recovered - Server 3  
407 - recovered - Server 2  
499 - recovered - Person D

**exp 2.6 - persistence = 45**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
74 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
208 - recovered - Server 1  
210 - infected - Server 3  
240 - recovered - Person B  
254 - recovered - Person C  
311 - infected - Person D  
373 - recovered - Server 2  
391 - recovered - Server 3  
492 - recovered - Person D

**exp 2.7 - persistence = 60**

21 - infected - Person A  
31 - infected - Server 1  
60 - infected - Person B  
71 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
211 - recovered - Server 1  
214 - infected - Server 3  
240 - recovered - Person B  
251 - recovered - Person C  
274 - infected - Person D  
373 - recovered - Server 2  
394 - recovered - Server 3  
455 - recovered - Person D

**exp 2.8 - persistence = 90**

21 - infected - Person A  
31 - infected - Server 1  
60 - infected - Person B  
67 - infected - Person C  
104 - infected - Server 2  
201 - recovered - Person A  
210 - infected - Server 3  
211 - recovered - Server 1  
240 - recovered - Person B  
248 - recovered - Person C  
267 - infected - Person D  
284 - recovered - Server 2  
391 - recovered - Server 3  
447 - recovered - Person D

## D.3 Contact rate

**exp 3.1 - contact rate = 0**

21 - infected - Person A

201 - recovered - Person A

**exp 3.2 - contact rate = 0.2**

21 - infected - Person A

201 - recovered - Person A

**exp 3.3 - contact rate = 0.4**

21 - infected - Person A

27 - infected - Server 1

72 - infected - Person C

78 - infected - Person B

201 - recovered - Person A

208 - recovered - Server 1

214 - infected - Server 3

252 - recovered - Person C

258 - recovered - Person B

394 - recovered - Server 3

**exp 3.4 - contact rate = 0.6**

21 - infected - Person A

27 - infected - Server 1

67 - infected - Person B

83 - infected - Person C

193 - infected - Server 2

201 - recovered - Person A

208 - recovered - Server 1

210 - infected - Server 3

247 - recovered - Person B  
264 - recovered - Person C  
311 - infected - Person D  
373 - recovered - Server 2  
391 - recovered - Server 3  
492 - recovered - Person D

**exp 3.5 - contact rate = 0.8**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
68 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
208 - recovered - Server 1  
214 - infected - Server 3  
240 - recovered - Person B  
249 - recovered - Person C  
311 - infected - Person D  
373 - recovered - Server 2  
394 - recovered - Server 3  
492 - recovered - Person D

**exp 3.6 - contact rate = 1.0**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
68 - infected - Person C  
193 - infected - Server 2  
201 - recovered - Person A  
208 - recovered - Server 1  
210 - infected - Server 3  
240 - recovered - Person B  
249 - recovered - Person C  
311 - infected - Person D  
373 - recovered - Server 2  
391 - recovered - Server 3

492 - recovered - Person D

## D.4 Recovery rate

### **exp 4.1 - recovery time = 30**

21 - infected - Person A  
27 - infected - Server 1  
51 - recovered - Person A  
58 - recovered - Server 1  
60 - infected - Person B  
72 - infected - Person C  
90 - recovered - Person B  
102 - recovered - Person C

### **exp 4.2 - recovery time = 60**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
76 - infected - Person C  
81 - recovered - Person A  
87 - recovered - Server 1  
120 - recovered - Person B  
136 - recovered - Person C

### **exp 4.3 - recovery time 90**

21 - infected - Person A  
31 - infected - Server 1  
62 - infected - Person B  
87 - infected - Person C  
111 - recovered - Person A  
121 - recovered - Server 1  
153 - recovered - Person B  
177 - recovered - Person C



210 - infected - Server 3  
300 - recovered - Server 3

**exp 4.4 - recovery time 120**

21 - infected - Person A  
31 - infected - Server 1  
60 - infected - Person B  
76 - infected - Person C  
141 - recovered - Person A  
151 - recovered - Server 1  
180 - recovered - Person B  
193 - infected - Server 2  
196 - recovered - Person C  
217 - infected - Server 3  
313 - recovered - Server 2  
322 - infected - Person D  
337 - recovered - Server 3  
442 - recovered - Person D

**exp 4.5 - recovery time 240**

21 - infected - Person A  
27 - infected - Server 1  
60 - infected - Person B  
68 - infected - Person C  
214 - infected - Server 3  
261 - recovered - Person A  
267 - infected - Person D  
268 - recovered - Server 1  
300 - recovered - Person B  
308 - recovered - Person C  
311 - infected - Server 2  
454 - recovered - Server 3  
508 - recovered - Person D  
552 - recovered - Server 2