A software toolkit for radio frequency data terminals

Andrew Buckingham

A Software Toolkit for Radio Frequency Data Terminals

M.Sc. Thesis

Andrew Buckingham
School of Engineering
University of Durham

Summary

This thesis is concerned with the computer industry of "automatic identification." Central to automatic identification is the use of barcodes and related technologies, including wireless computer networks and Radio Frequency Data Terminals (RFDTs)

RFDTs are hand held computers incorporating a screen, barcode scanner, and radio transceiver. Programming of these devices is bespoke to each customer, hence this thesis' subject: producing a toolkit for economical development of RFDT software.

The work reported in this thesis was developed in a joint project between Castle Auto I.D. Solutions[1], a barcode systems integration company, and the University of Durham. This was enabled by a TCS Programme[2], which places a university employee within the company for "technology transfer."

The thesis begins with a study of the relevant hardware and technology in the automatic identification industry. The common software components of data collection systems are then identified, and a "toolkit" of reusable software components is proposed, implemented, and refined. Particular attention is paid to code re-use. In refining the toolkit, several commercial applications are developed and deployed into industry.

The final case study is a commercially successful 802.11b wireless site survey system, developed using the toolkit.

Acknowledgements

The author would like to thank the following people and companies, for their professional support during this thesis:

Castle Auto I.D. Solutions, for funding and facilitating the work. Thanks to Kevin Brown, Derek Stainton and Janet Stables.

TCD (Teaching Company Directorate), now known as KTP, for funding; in particular to Richard Parker-Smith for his support.

Jim Swift at the University of Durham, for continued supervision and encouragement.

On a personal note, the author also thanks:

Mum, Paul, Elizabeth Swift, Peter Baxendale, and Beccy.

Table of Contents

Index Of Figures

Nomenclature

AP	Access Point. Converts wired Ethernet to wireless Ethernet signals.
CCD	Charge Coupled Device. A semiconductor which converts light to electrical signals. Can be arranged in a linear or two-dimensional array.
CCK	Complementary Code Keying. A particular type of encoding used in DSSS.
COM	Component Object Model. Microsoft's standard for cross-language code library calls.
CR	Carriage Return character. Equivalent of pressing Return/Enter.
DSSS	Direct Sequence Spread Spectrum. Type of radio modulation for wireless Ethernet 802.11 and 802.11b
ESSID	Extended Service Set ID. The network name of a wireless Ethernet network, managed by an Access Point.
FHSS	Frequency Hopping Spread Spectrum. Type of radio modulation for wireless Ethernet 802.11.
HF	High Frequency radio signals.
IDL	Interface Description Language, for describing a COM Interface.
IP	Internet Protocol. A data routing protocol used for computer networks, used in TCP/IP.
ISM	Industrial Scientific and Medical. The 2.4GHz licence-free radio band.
ITM	Inter-Thread Message. Method of communication between different threads.
LED	Light Emitting Diode. A semiconductor which emits light.
Mbps	Megabits per second. Measure of data transfer speed, in millions of bits per second.
MFC	Microsoft Foundation Classes. C++ class library and accompanying binary code library, providing access to many functions of Windows.
MT	Multi-threading / Multitasking: ability of an operating system to run several programs apparently simultaneously.
Narrowband	RF transmission on lower frequencies (e.g. 433MHz) at low data rates.
OS	Operating System. Low level computer software which provides access to peripherals, memory, and CPU. User software runs under the control of the OS. Microsoft Windows 2000 is an example of an OS.

PCMCIA	Format of PC expansion card, often used for implementing network cards. PDAs and PC laptops can have PCMCIA slots.
PDA	Personal Digital Assistant. Consumer hand held computer.
PDT	Portable Data Terminal. Industrial hand held computer.
QPSK	Quadrature Phase Shift Keying. Method of encoding two bits of data onto an RF carrier, by shifting the carrier phase through multiples of 90 degrees.
RF	Radio Frequency transmission.
RFDT	Radio Frequency Data Terminal. PDT with RF capabilities.
RS232	Communications protocol for connecting devices to a PC. Can be used by barcode scanners.
SDK	Software Development Kit. Software tool to assist developers in writing code. SDKs usually include extensive code libraries, providing programmatical access to OS functions.
SNR	Signal to Noise Ratio, with reference to RF transmissions.
TCP/IP	Standard for encapsulating data for transport over a computer network.
TELNET	Standard network port over which Video Terminal (VT) codes are sent. For interacting with a VT compatible client computer.
VT	Video Terminal: standard for defining how to control a VT compatible client computer, by sending "control codes" in the data stream to format the screen.
WEP	Wired Equivalency Privacy. A kind of encryption for wireless data packets, which gives a level of security against unauthorised access.
WMS	Warehouse Management System. A computer system for controlling stock movements, often using wireless hand-held computers with barcode readers.

1 Introduction

This thesis discusses data collection technology, which is used widely within industry. A study will first be made of the commercial background, before moving on to a detailed examination of data collection and wireless radio frequency (RF) systems.

1.1 Stock Control using Real Time Data

All disciplines of manufacture and distribution require feedback of information. This might be for stock levels in a distribution centre, counting items out; or it may be for tracking items as they move through the stages of a production facility.

One advantage of having real-time, up to the minute data, is improved stock control. If a process can keep stock to a minimum, then cash flow is improved in the company.

For example, a supermarket can store a minimum of stock in the warehouse, to minimise the time which goods are standing within store. The shelves are replenished on a just-in-time basis, with new goods arriving only when the shelves are about to become empty.

To allow such critical timing in distributing goods, gathering data on stock movements must be done sufficiently frequently and with very little delay.

Gathering these kinds of data is a common requirement in industry, and usually employs wireless barcode readers which connect directly to a central computer system.

This thesis shall concentrate on the integration of those wireless barcode readers into software systems. Other peripheral components of wireless data collection will also be considered.

1.2 Portable RF Data Terminals

There are several terms for these portable devices. The broadest umbrella is perhaps Portable Data Terminal. These devices all have some method of sensing and storing data, and of displaying feedback to the operator.

However, one more factor must be considered against the background of the process: how quickly must data be fed back.

Some processes will only require feedback once per day, or per hour, or some other period of batch processing. In this case, the operator will return the device to a basestation where the collected information can be processed.

Other processes may require the data to be made instantly available to the system, as soon as it is collected. This is where Radio Frequency Data Terminals (RFDTs) may be used: devices which have all the above attributes, but with the benefit that they are connected via a radio network to the system.

It is these RFDTs on which this thesis will concentrate: their abilities, features, and integration into a manufacturing or distribution system.

1.2.1 RFDT Form Factor

Physically, the RFDT is a hand-held unit including the following key features:

- Screen to give feedback to the user
- Keypad to control the RFDT and supply user input
- Barcode reader
- Radio transceiver

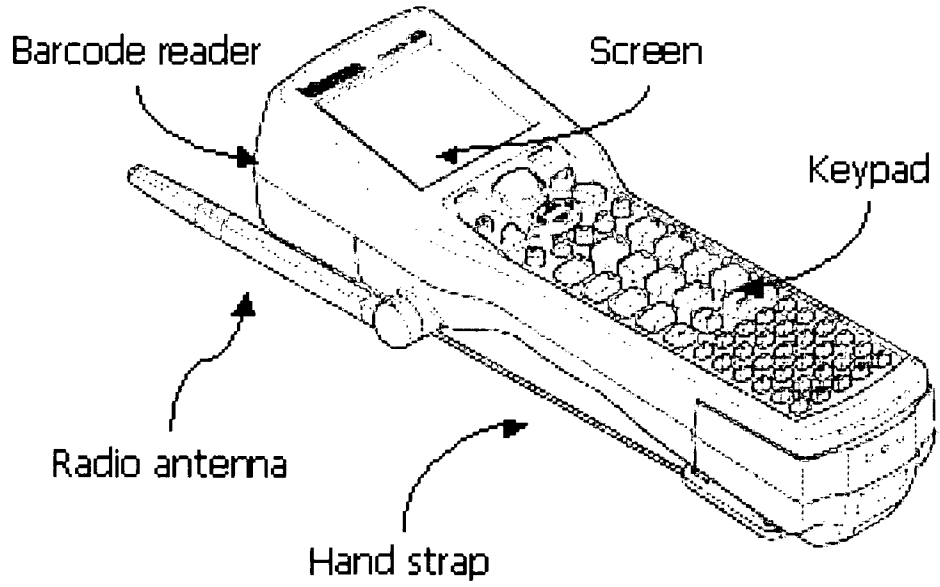

Figure 1: RFDT outline

1.3 System Configurations

As detailed in the previous section, many manufacturing or distribution processes will require real-time feedback of information to a computer. This will enable a database of some sort to store the information, process it, and return information to other devices or to the operators.

There will be several types of component in this system:

- Collection devices: RFDTs
- Network infrastructure: Hubs, switches, routers, wireless networks
- Processing: Servers, PCs, RFDTs
- Outputs: RFDTs, printers, PCs

1.4 Warehouse Management System Example

In order to demonstrate how an RFDT may be included in a system, a Warehouse Management System (WMS) will be considered. WMS are a good example of using RFDTs within industry.

A WMS controls the flow of goods into and out of a warehouse. Many older warehouses may still use paper-based systems, whereas newer ones may use a database to record where the goods are.

However, all WMS require the following operations:

1.4.1 Goods In

As each batch of product is brought in, it will be verified against a list of expected incoming goods, and then placed in the appropriate location within the warehouse.

1.4.2 Stock Checking

Periodically, it is necessary to verify the contents of the warehouse against the expected stock levels.

1.4.3 Goods Out

When products are removed to be distributed to a customer, they must also be removed from the stock list.

1.4.4 Goods In Example

The goods in operation will be examined in detail, as an example of the advantages of RFDTs over paper-based systems.

1.4.4.1 Paper Based Operation

A list of products in each batch will be given to the worker. This list may take a form similar to the following:

1. Print out a list of the expected goods in a batch or pallet.

2. Take a product from the pallet.
3. Place the product at the appropriate location.
4. Tick off the product on the list, recording how many were received.
5. If there are more products on the pallet, go back to stage 2.
6. Update central records according to which items were ticked on the list.

The following diagram shows how such a list might look to the operator.

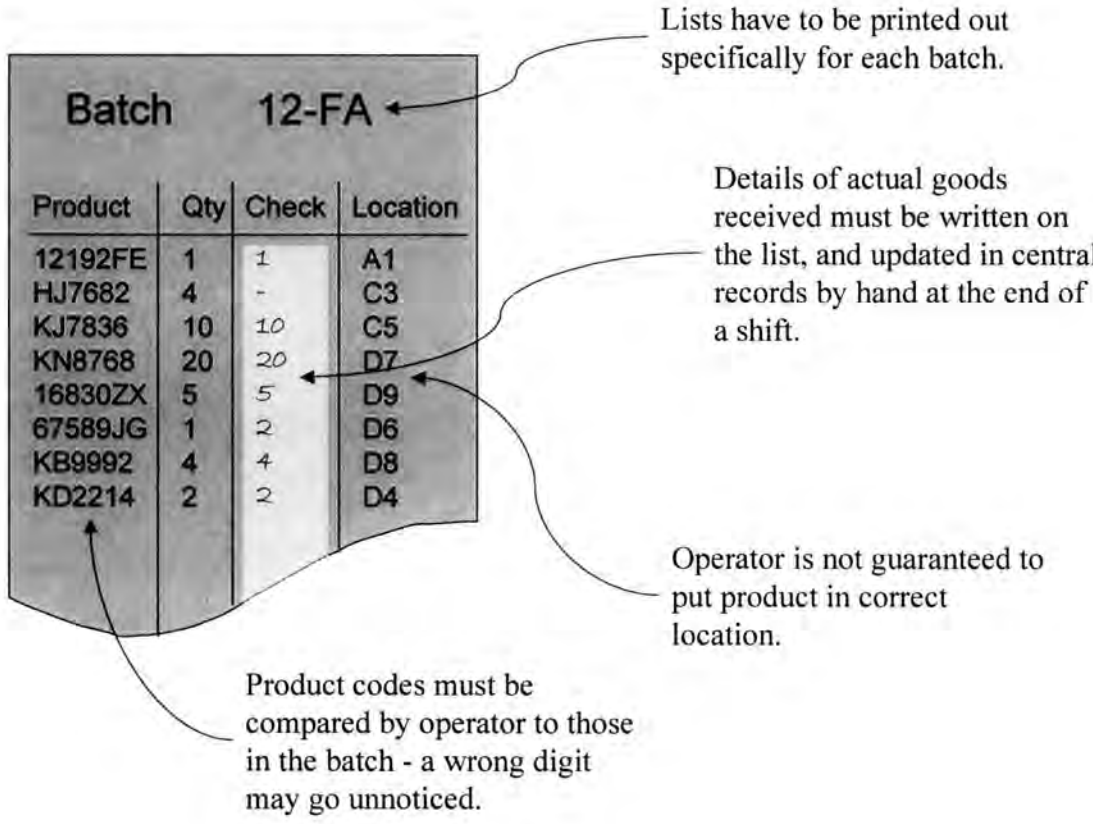

Figure 2: Manual goods-in

1.4.4.2 RFDT Operation

Instead of filling in details on a list, the operator will be prompted at every stage of the process by the RFDT.

The procedure would typically be as follows:

1. Operator scans barcode of batch.

2. List of expected goods appears on screen.
3. Operator scans barcode of any product in the batch.
4. Operators enters quantity received.
5. RFDT displays which location to place the product at.
6. Operator scans location when the product is placed there, to confirm.
7. If there are more products on the pallet, go back to stage 3.

The main advantages of the RFDT are:

- There less room for human error with data entry, as all products and locations are barcoded. E.g., a product cannot be placed at the wrong location, as the scanned location barcode would be rejected by the RFDT.
- There is no need to update central records manually from each list, as the RFDT talks directly to a database when each product is stored.
- The paper lists of each batch are not required, so there is no paper to be lost as products are moved around the warehouse.
- The time of each operation can be recorded, which means that operators' productivities can be assessed more easily.

The on-screen prompts would look similar to the following:

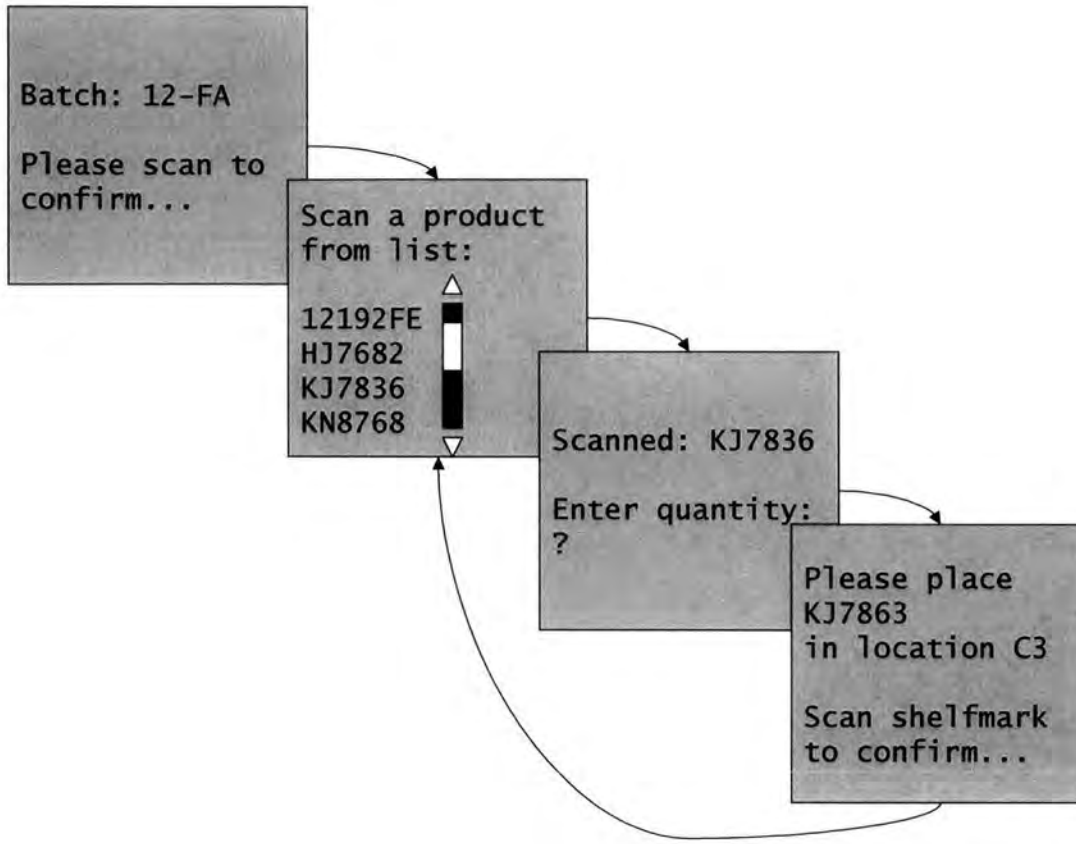

Figure 3: RFDT goods-in

This concludes the WMS example, demonstrating the usefulness of RFDTs in such a situation.

2 Technology Review

There is a broad range of technology in any RFDT based system. This includes barcoding, networking, wireless communications, and programming. These will be discussed in this technology review.

2.1 Barcoding Technology

A central theme to WMS automation is the barcoding of products. This enables data to be put into the system very rapidly, without the large amounts of time and mistakes which operators can make during input.

There are many types of barcode encoding, or "symbology", and also several types of barcode reader.

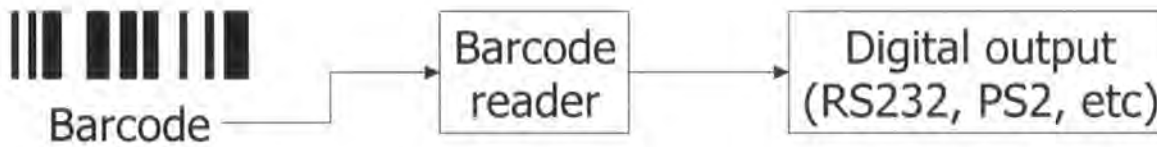

Figure 4: Barcode reading block diagram

RS232 and PS2 are just two examples of computer interface. RS232 is sometimes known as serial, and PS2 is the common keyboard interface to a computer.

An outline of the technology is as follows.

2.1.1 Barcode Symbologies

There are many types of symbology, and most people will encounter several types during their day-to-day business. Some types are indistinguishable from others, and some barcodes are hardly recognisable as barcodes at all.

Different symbologies exist to serve different purposes. A symbology is selected depending upon the following requirements:

- Data density (characters stored in a given area)

- Character set encoding (e.g. all ASCII chars, or just numerically 0-9)
- Type of barcode reader required (laser, LED, CCD imager – see later sections)
- Redundancy (part of the barcode can be destroyed, yet no data is lost)
- Integrity (self validation by checksum)
- Tolerance to printing inaccuracies

This section will classify the symbologies into their main groups, giving examples of each. Broadly speaking, the symbologies fall into one of two categories:

- One-dimensional (1D) symbologies
- Two-dimensional (2D) and stacked symbologies

2.1.1.1 One-Dimensional Barcodes

One-dimensional (1D) barcodes are the most easily recognisable to the general public. For example, they are used on all supermarket goods to identify the products at the checkout till.

Figure 5: 1D barcode examples

All 1D barcodes are a combination of the following parameters[3]:

- Bar: the dark elements
- Space: the light elements
- Element: generic term for a bar or space

- Inter-character gap: "discrete" symbologies leave a space between the last bar of a character and the first bar of the following character
- Module: the narrowest base element, of which all bars and spaces are a multiple
- X dimension: the module width
- Quiet zone: the margin extending to the left and right of the barcode, where there must be no elements
- Human readable: the encoded characters may be reproduced next to the barcode, so that humans can read the content visually

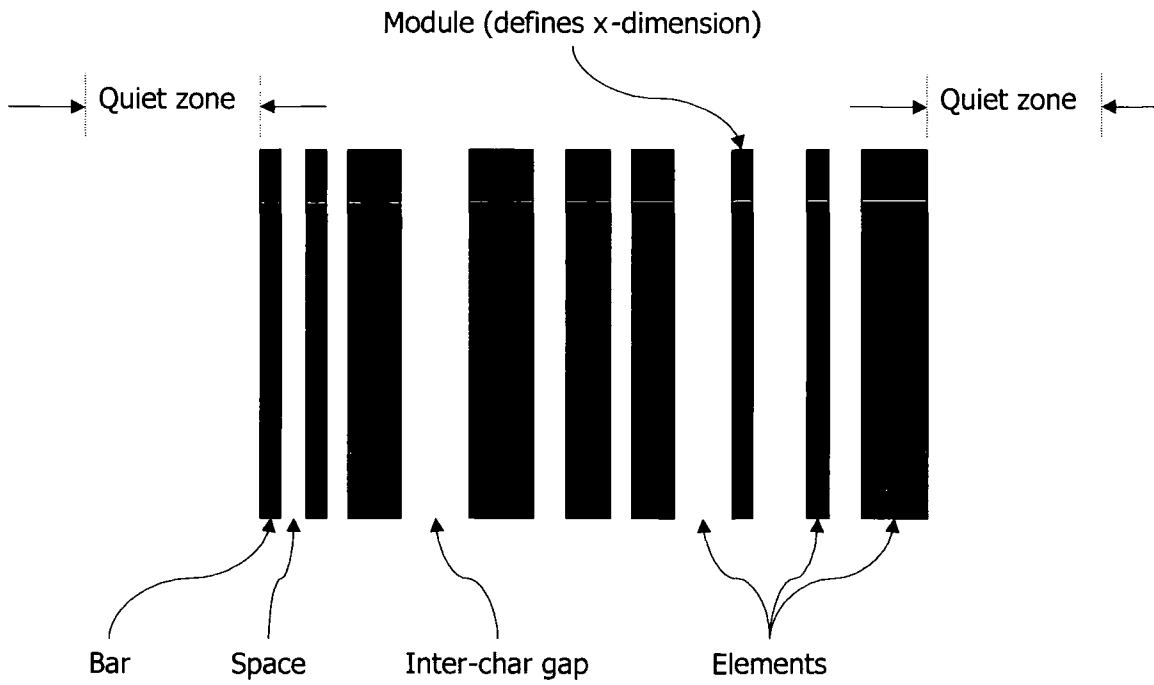

Figure 6: 1D barcode parameters

The following table details the relative merits of some popular 1D symbologies.

	EAN-13	**2/5 standard**	**2/5 interleaved**	**Code 39**	**Code 128**
Characters encoded	0 to 9	0 to 9	0 to 9	Alphanumeric	Full 7-bit ASCII char set
Checksum	Yes	Yes	Yes	Yes	Yes
Max length	13 digits	Variable	Variable	Variable	Variable
Typical application	POS	Transport industry	Transport industry	Electronics industry	Logistics & distribution
Advantages	High data density	Good print tolerance	High data density	Alphanumeric	Full ASCII, high data density
Disadvantages	Low tolerance	Low data density	Low print tolerance	Low data density	Low print tolerance

Table 1: Comparison of 1D symbologies

2.1.1.2 Detailed Description of Code 128

Code 128 is a popular 1D symbology, due to having both a high data density and the full ASCII character encoding. This section will describe how each character is encoded.

Referring back to Figure 6, Code 128 parameters are as follows:

Parameter	Value
Modules per character	11
Bars per character	3
Spaces per character	3
Total for bars in each character	Even number of modules
Total for spaces in each character	Odd number of modules
Possible element widths	1, 2, 3 or 4 modules

Table 2: Parameters for Code 128

This is easier to comprehend when some example character encodings are considered. The following table gives the width of each of the bars (B1-B3) and spaces (S1-S3), in modules, for some example characters.

Char	B1	S1	B2	S2	B3	S3	Total bar modules	Total space modules	Total char modules
"A"	1	1	1	3	2	3	4	7	11
"B"	1	3	1	1	2	3	4	7	11
"8"	3	1	1	2	2	2	6	5	11

Table 3: Code 128 encoding example

Given the above rules for how each character is encoded, it is possible to calculate the maximum number of distinct characters as being 126.

Code 128 achieves such a high data density by having 4 variations of element width, and by encoding data in the spaces as well as the bars. This is in contradiction to simpler code types such as Code 39, where no data is encoded in spaces.

However, the cost of this high data density and complexity is the requirement for accurate printing.

2.1.1.3 Two-Dimensional Barcodes

Whereas in 1D barcodes the data is encoded only left-to-right, 2D symbologies store data top to bottom as well. An analogy can be drawn with written English: 1D barcodes are like single lines of text, whereas 2D barcodes are akin to whole paragraphs.

The main advantage of this is the ability to encode large amounts of data in a single barcode: up to several kilobytes in extreme cases.

A broad outline of the types of 2D barcodes is as follows.

- Stacked barcodes (structured like 1D codes printed one above another)
- Matrix barcodes

2.1.1.4 Stacked Barcodes

These are visually just several 1D barcodes which are literally stacked one above the other, with a common start and end marker surrounding them.

For example, "Codablock F" consists of up to 44 rows of Code 128, each line being up to 62 characters, giving a total theoretical capacity of 2.7KB.

However, the more widely used PDF417 stacked barcode uses its own unique structure for each line:

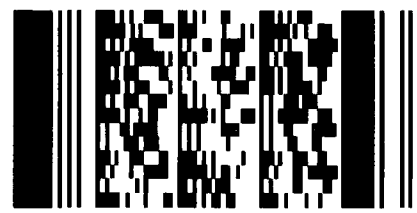

Figure 7: Stacked barcode example (PDF417)

It can be seen in the above example that each line looks like a traditional 1D barcode, with a larger, outer 1D barcode surrounding the inner lines. The outer barcode is used as a general start-stop indicator for the inner lines.

2.1.1.5 Matrix Barcodes

The general public would probably not recognise these as being barcodes; such is their difference from 1D and stacked symbologies.

A grid of black and white elements is arranged in a complex pattern, giving high data density and built in redundancy. The redundancy means that a certain percentage (definable at design time) of the barcode may be completely destroyed, and yet all the data can still be reconstructed by the reader.

The two main types of matrix symbology are:

- Data Matrix, rectangular array
- Maxi Code, hexagonal array

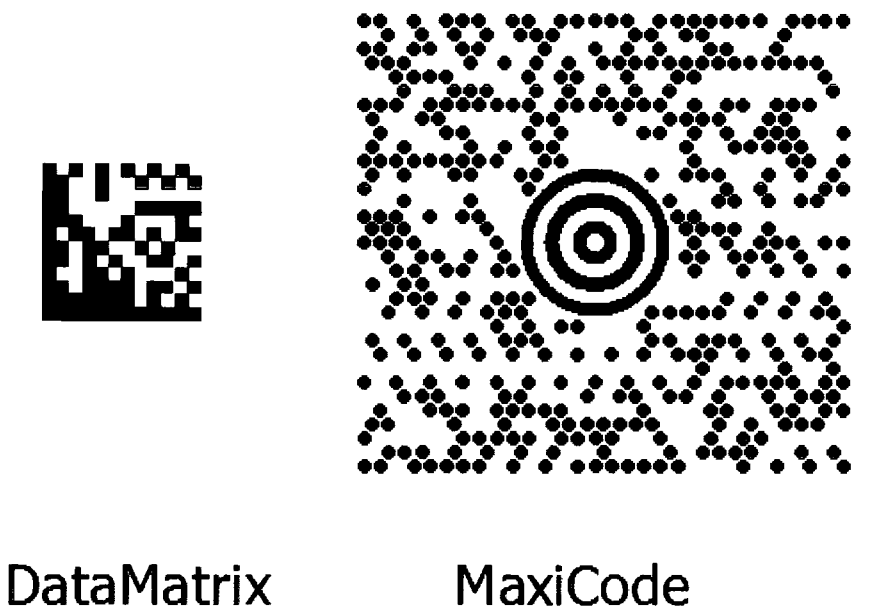

Figure 8: 2D barcode example (DataMatrix and MaxiCode)

These have the following characteristics[4]:

	Data Matrix	**Maxi Code**
Structure	Rectangular array	Hexagonal array
Orientation markers	Border lines	Bullseye
Max grid size (elements)	144 x 144	33 x 30
Max capacity (ASCII chars)	2,334	93
Max size (mm x mm)	Variable	25 x 25 (fixed)
Max redundancy	25%	25%

Table 4: Matrix symbologies comparison

2.1.2 Barcode Readers

With so many types of barcode symbology, it is of no surprise that there are also different types of barcode reader. This section will introduce the main groups, and detail the operation of the popular readers:

- Pen
- Linear LED
- Linear laser
- Omni-directional laser
- CCD imager

2.1.2.1 Pen

The pen style of reader is the most basic, and is rarely seen. It requires the operator to run the pen along the barcode, and detects the pattern of light and dark reflections. It is slow to use, and requires the operator to take care when scanning. Only 1D symbologies can be read.

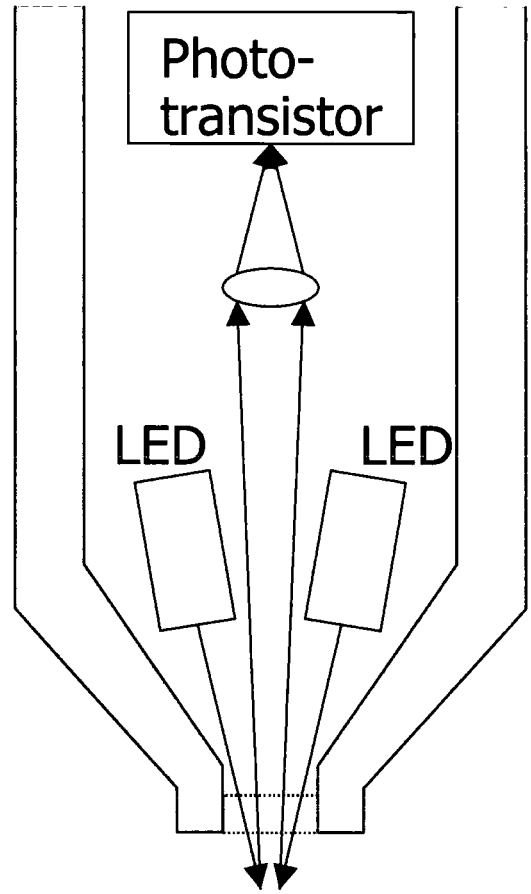

Figure 9: Pen reader

2.1.2.2 Linear LED/CCD

These are the most popular and most robust form of reader. They use an array of LEDs (light emitting diodes) to illuminate the barcode, and have a linear array of CCDs (charge coupled devices, i.e. light intensity detectors) to distinguish the dark areas from the light areas.

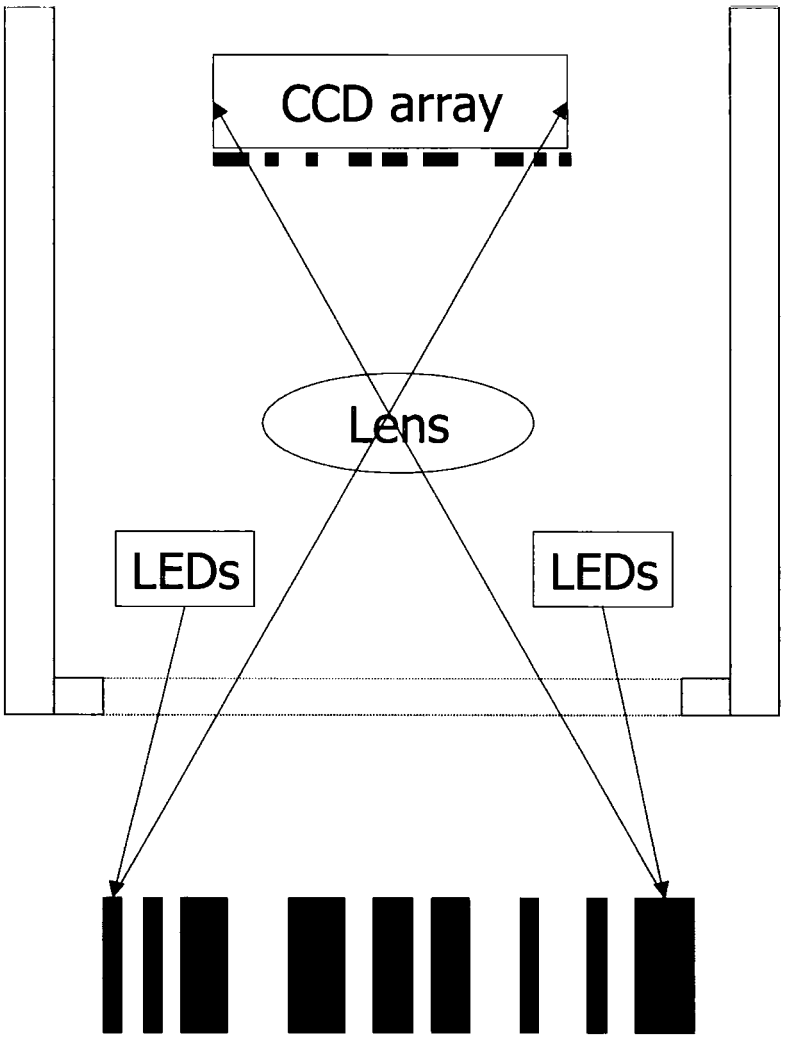

Figure 10: Linear LED/CCD reader

These are best suited to 1D barcodes; however, some advanced models allow the operator to scan down a stacked barcode to interpret Codablock and PDF417.

2.1.2.3 Linear Laser

Laser scanners use a rapidly oscillating or rotating mirror to raster a laser spot into a scan line. The reflected laser spot is detected by a photo detector, which thus produces a voltage which varies in sympathy with the reflected light from the barcode.

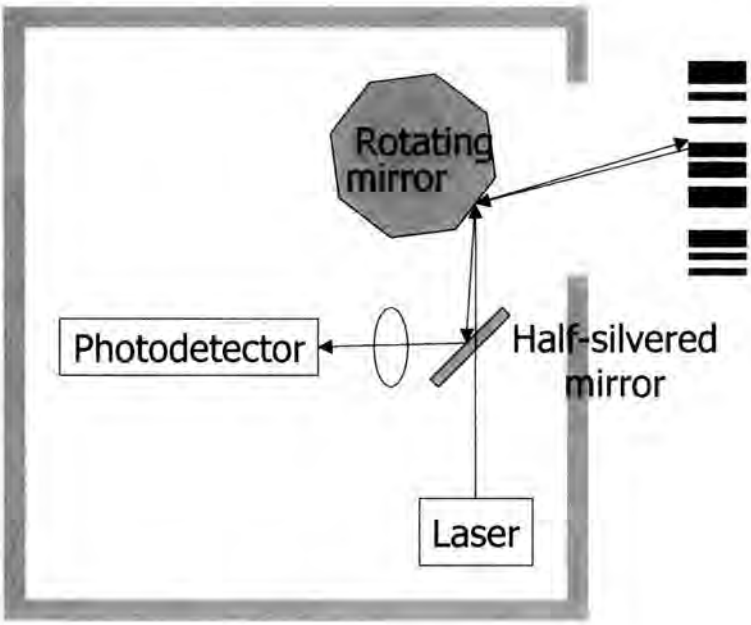

Figure 11: Laser reader

Laser scanners, as with LED CCDs, are best suited to 1D or occasionally stacked barcodes. However, lasers have the advantage of a very long range, due to the reduced divergence of laser light when compared with LED illumination.

2.1.2.4 Omni-Directional Laser

In addition to a mirror rastering the laser left and right, a further direction of movement rasters this line into a series of lines. This scan pattern can be quite complex, and allows the barcode to be at virtually any orientation and distance.

These are commonly in supermarket checkouts, due to their high speed of operation, and ability to read at a wide range of orientations.

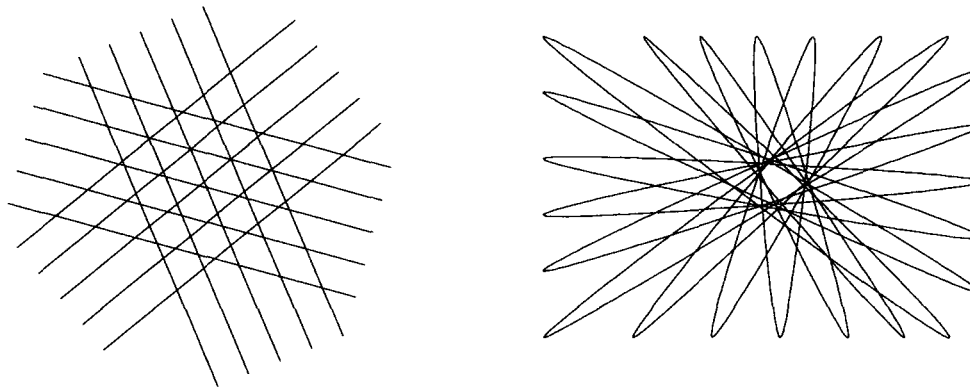

Figure 12: Example omni-directional scan patterns

2.1.2.5 CCD Imager

Based on a CCD camera and an embedded microprocessor, the CCD imager photographs the barcode and performs image analysis to decode the symbology.

The key advantages of this system are that any symbology can be read, at any angle, at a great range of distances. A 2KB Maxicode barcode can be read and interpreted in less than one second.

However, a reasonably powerful microprocessor and high-resolution CCD make for an expensive item.

2.1.2.6 Comparison of Reader Types

	Pen	**LED**	**Linear laser**	**Omni laser**	**CCD imager**
Max. read distance	Contact ★	Medium ★★	Far ★★★	Far ★★★	Far ★★★
1D barcodes	Manual ★★	Auto ★★★	Auto ★★★	Auto ★★★	Auto ★★★
Stacked barcodes	No	Manual ★	Manual ★	No	Auto ★★★
2D matrix barcodes	No	No	No	No	Auto ★★★
Moving parts	None ★★★	None ★★★	Some ★★	Many ★	None ★★★

Table 5: Reader type comparison

The ratings used above are from one to three stars.

2.1.2.7 Reader Types in RFDTs

Because RFDTs are aimed at the high-end of the barcode technology market, they usually have either a linear laser or, increasingly, a CCD imager.

2.2 Networks

It has been described that RFDTs must fit into a complete system, and integrate with other computers on a network. This section will outline what a network is, with specific reference to 802.3 wired Ethernet and 802.11b wireless Ethernet.

2.2.1 Purpose

The network must move information between computers, fulfilling the following criteria:

- Adequate data transfer speed;
- Reliability;
- Broad interconnectivity;
- Resilience to errors;
- Data security.

There are many different standards for computer networks, most of which will not directly connect to each other. However, devices called "routers" can assist in connecting different network types together.

Before working out how to connect two different network types, each type of network must first be formalised so that they can be directly compared. One formal framework for defining a network is known as the OSI (Open Systems Interconnection) 7-layer model[5].

2.2.2 OSI 7-layer Model

Most common network types in use today can be fitted into the 7-layer model:

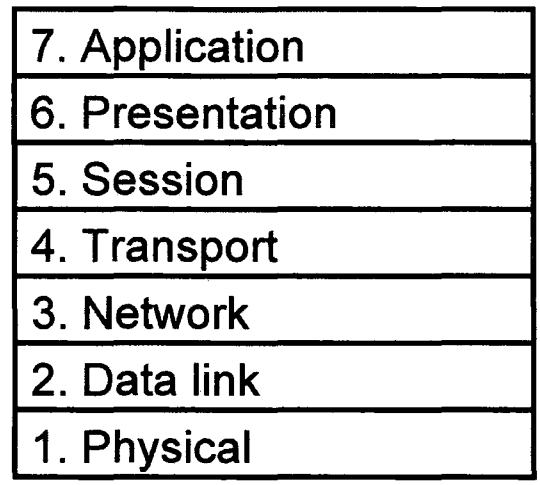

Figure 13: OSI 7-layer model

Each layer performs a function which is abstracted from the layer built on top of it, so that higher layers (e.g. transmitting a data file) need not be concerned with the lower-level functions (e.g. modulating the data bits onto a physical signal).

These layers will be discussed below.

2.2.2.1 Layer 1 – Physical

This layer defines how raw data bits are physically transmitted across a communications channel. Examples of physical transmission include an electrical voltage (e.g. 802.3 wired Ethernet), Infra-red, and 2.4GHz radio frequency. The length of time for which a bit lasts is also defined here.

2.2.2.2 Layer 2 – Data Link

Transmission errors are detected at this layer, for example electrical noise on a line might change a bit value. As part of this error detection process, the Data Link layer splits data into "frames" of a few hundred or thousand bits. Special bit patterns can be appended or prefixed to the frames, to allow the Data Link layer to check for errors.

Acknowledgement frames are also sent and processed, so that frames which do not arrive at the destination are re-broadcasted.

Data Link includes the Medium Access Control (MAC) sub-layer, in networks where a data line is shared between many network devices. The MAC controls how those

devices share the same line: if two or more devices talk simultaneously, they are indistinguishable and appear as noise—a "collision"—so this must be avoided.

2.2.2.3 Layer 3 – Network

In order to connect large numbers of computers, a network may be split into different segments (or "subnets"). The switching or routing of packets between subnets is handled by the network layer.

The Network layer also handles interconnecting different types of network, which may have different protocols, maximum data packet lengths, or different addressing systems.

One common addressing system, which is routed by the Network layer, is Internet Protocol (IP). This has addresses in the format of 4 eight-bit bytes, separated by full stops. I.e. ranging from 0.0.0.0 to 255.255.255.255

2.2.2.4 Layer 4 – Transport

The transport layer splits up data into packets before passing them to the Network layer. The exact method of packet transmission is hidden by the Network layer (i.e. the packet format), and packets may go through several different types of network which are hidden to the Transport layer.

However, the Transport layer can provide other functions such as: ensuring packets are re-assembled in the correct order at the destination, if the Network layer routes them non-sequentially; allowing multiple connections per computer and deciding which packets are for which connection; and flow control to ensure that the destination computer has enough time to process packets before new ones arrive.

2.2.2.5 Layer 5 – Session

The Session layer builds slightly upon Transport. It controls whether packets flow in both directions simultaneously, or take turns using a token relay system. It can also provide "checkpoints" in a session, so that the session can be re-opened following a system crash and resume from an agreed data packet number.

2.2.2.6 Layer 6 – Presentation

Although all computers use data bytes to represent information, their methods of encoding information in bytes and bits can differ. The Presentation layer can define the semantics of how information is represented. For example, the standard method for encoding text strings can vary: some computers use ASCII (8 bits per character, so a max of 255 characters in the alphabet); some computers use Unicode (16 bits per character, or 65535 characters).

The Presentation layer can define which formats are used for the data transmission.

2.2.2.7 Layer 7 – Application

This is the very top level, and defines the functions which are performed on a network. Typical examples are File Transfer Protocol (FTP), Windows Networking, Telnet, and HTTP (i.e. World Wide Web).

2.2.3 TCP/IP

The most common type of networking used for PCs and RFDTs is TCP/IP. This is the networking which is used across the Internet. TCP/IP connections are commonly referred to as "Sockets."

TCP/IP does not map perfectly to the OSI model, but it can be approximated as in the following diagram[6]:

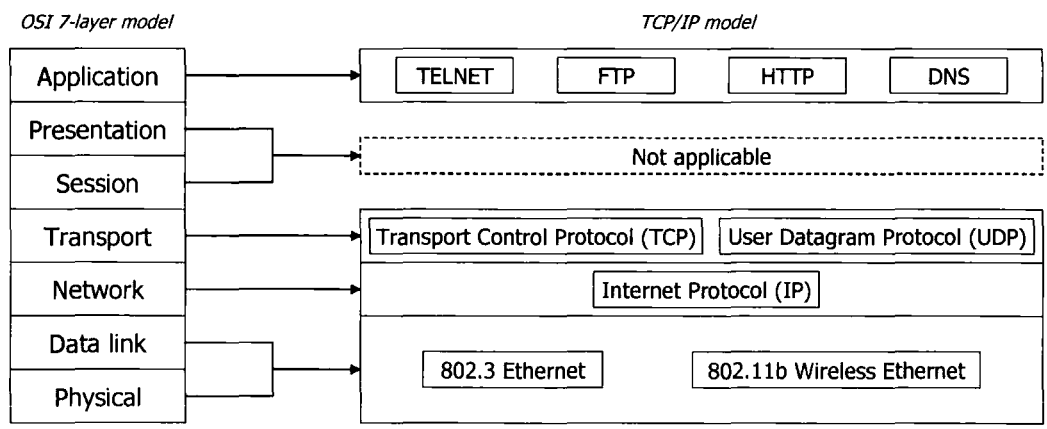

Figure 14: TCP/IP in the OSI 7-layer model

2.2.4 Hubs

A network hub duplicates information across several network connections:

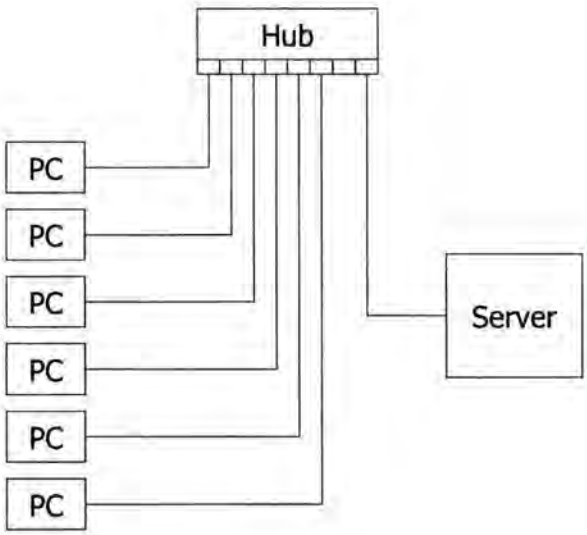

Figure 15: Using a hub

In the above diagram, a number of PCs are all linked to the server. A signal from any one network device is duplicated across all the other network connections.

Basic hubs operate at layer 1 of the OSI model.

2.2.5 Switches

These are a more intelligent form of hub. Switches also link multiple network devices together, but instead of just reproducing the signal across all connections, they examine the data packets to see which device they are destined for. Depending upon the destination address, the switch only sends the data to the appropriate network device.

Switches operate up to layer 3 of the OSI model. They can examine IP packets to see what the destination computer address is, and send the data only on the path leading to the destination computer.

2.2.6 IEEE 802.3 Ethernet

"Ethernet" is a group of network types defined within IEEE standards[7]. Ethernet comes in several forms, but the type referred to in this thesis is 10/100BaseT. This refers to the physical wiring used; 8 core with an RJ-45 terminator.

2.2.7 Wireless Ethernet

This recent development in Ethernet replaces the physical wires with a 2.4GHz radio frequency (RF) carrier. It is designed to integrate with existing Ethernet systems.

Wireless Ethernet forms the backbone of an RFDT system, and this thesis expands upon it subsequently in section 2.3.

2.3 802.11 Wireless Ethernet

One of the reasons for the popularity of wireless Ethernet has been its adoption by IEEE (Institute of Electrical and Electronic Engineers), and its corresponding IEEE standard definitions.

This has encouraged interoperability between different manufacturers. Broadly speaking, all manufacturers ensure that their devices communicate with other manufacturers'.

Wireless Ethernet consists of two parts: the electromagnetic RF carrier, and the spread spectrum encoding technique. These will be discussed in sections 2.4 and 2.5 respectively.

A brief description of the different 802.11 standards is given below.

2.3.1 IEEE 802.11 Standard

This standard[8] was ratified in 1997, and uses layers[9] 1 and 2 of the OSI model (see section 2.2.2)

Layer 1 replaces[10] the Ethernet wire with 2402-2480 MHz, in either Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS) modulation. Layer 2 differs from the Ethernet MAC with a system more appropriate for wireless stations.

There are several classes defined within 802.11. These are 1 and 2 Megabits per second (Mbps) DSSS and FHSS, as well as Infra Red (IrDA). However, this thesis focuses on the DSSS.

2.3.2 IEEE 802.11b Standard

Two years after the ratification of standard 802.11 there was a modification; 802.11b. This is primarily the same as 802.11 DSSS, but the maximum data rate was increased[11] from 2 Mbps to 11 Mbps by modifying the physical (layer 1) definition. It is backward compatible with the existing 802.11 DSSS standard.

IEEE 802.11b has been widely adopted by several network hardware manufacturers, including: 3 Com, Agere, Avaya, Cisco, D-Link, Enterasys, Hewlett Packard, and Lucent (later forming Avaya and Agere, then purchased by Proxim.)

At the time of writing this thesis, 802.11b is the industry standard for both RFDTs and consumer PCs. 802.11b DSSS is described in a subsequent section.

2.3.3 IEEE 802.11g Standard

The 11g standard is still an emerging technology. It uses the same frequency range as 802.11b, and is backward compatible with that standard. However, data rates up to 54 Mbps are supported.

This thesis focuses on 802.11b, as 802.11g is not yet used in RFDTs.

2.3.4 IEEE 802.11a Standard

RF hardware based upon 802.11a was made commercially available at the same time as 802.11g (see above), and is also an emerging technology at the time of writing this thesis.

Like 802.11g, it promises higher data rates, at up to 54 Mbps. However, it uses a license band which is not wholly available in the UK, at 5 GHz. This is in contradiction to the other 802.11 standards, which all use license-free 2.4 GHz. The coverage range of 802.11a is less than 802.11b, due to the higher frequency RF having poorer penetration through materials.

This thesis focuses on 802.11b, as 802.11a is not yet used in RFDTs.

2.3.5 Basic Operation

Wireless Ethernet can operate in Peer-to-Peer mode ("Ad-hoc" or "Basic Service Set, BSS"[12]), where wireless client devices communicate amongst themselves. However, the preferred method of operation has a device called an Access Point (AP) controlling communication between clients ("Extended Service Set, ESS"[13]).

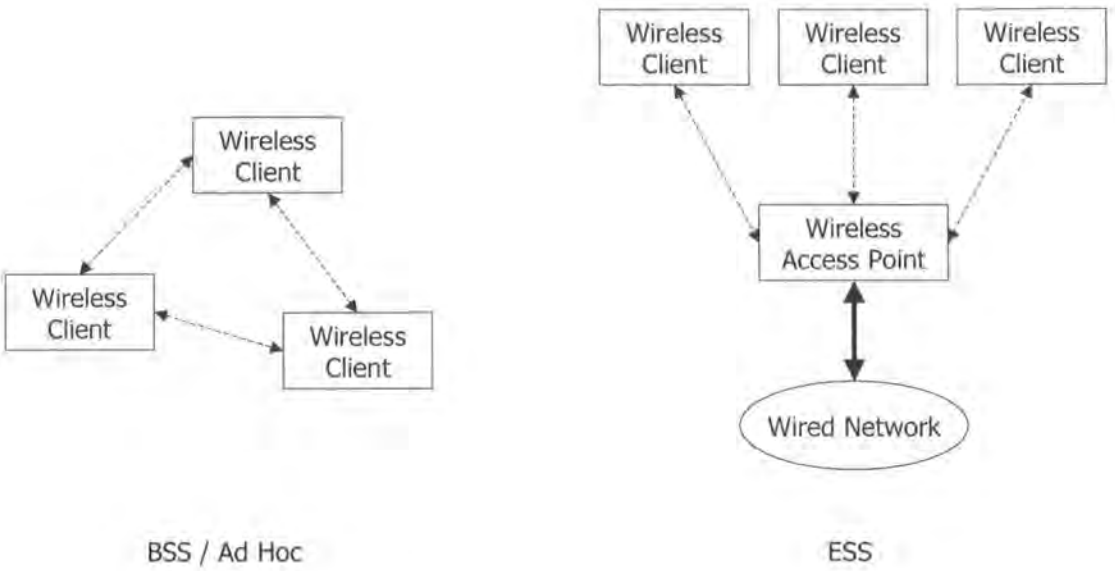

Figure 16: BSS versus ESS

ESS has two major advantages:

- Better data throughput through reduced collisions, because the Access Point (AP) controls when wireless client devices transmit;
- The wireless clients are bridged to the wired network, hence allowing connection to larger networks.

2.3.6 Hardware

As previously stated, a wireless system consists of access points (APs) and clients. In terms of hardware, the radio communication is usually encapsulated in a PCMCIA card format. As well as fitting into PDAs or RFDTs, the access points are also designed to use PCMCIA cards.

By including the PCMCIA cards in a range of devices, manufacturing cost is reduced through economies of scale.

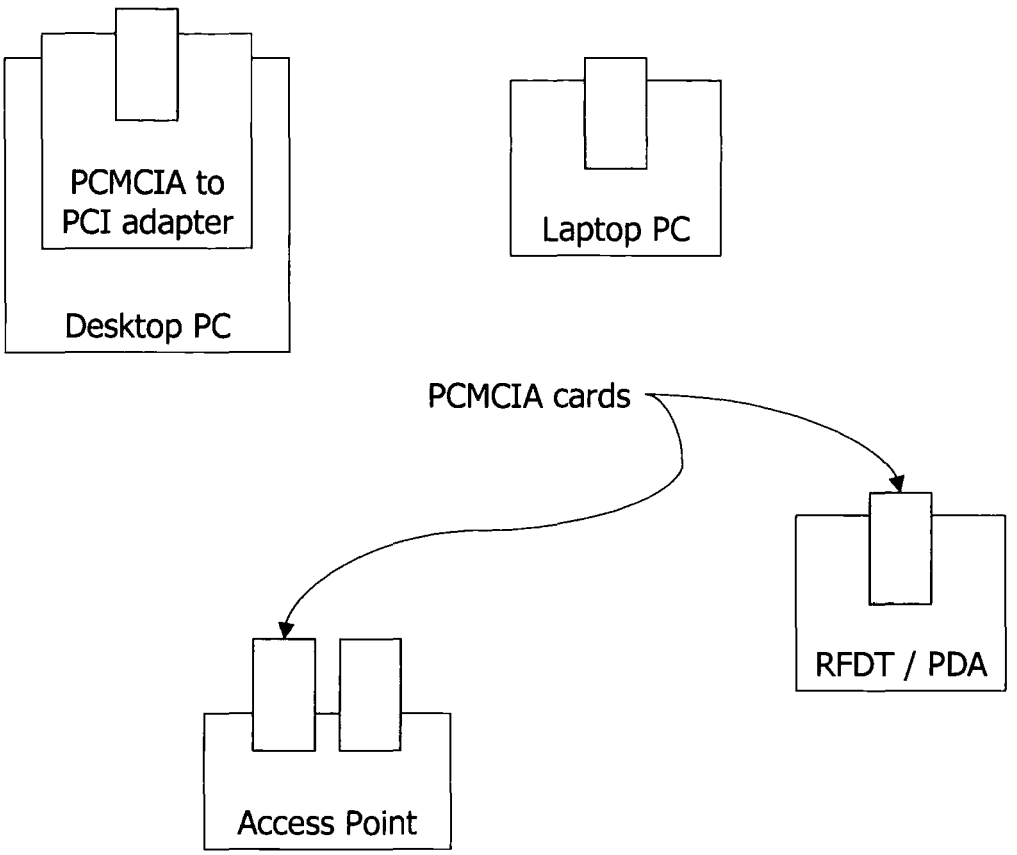

Figure 17: Wireless hardware

2.3.7 Configuration Parameters

From the user's point of view, there are three parameters to consider with wireless networking:

- Network Name (ESSID)
- Channel
- Encryption Key (WEP)

These parameters must match on both the client (e.g. RFDT) and the master device (i.e. wireless Access Point) if data transfer is to occur.

2.3.7.1 ESSID

It is possible that two separate companies may run wireless networks in adjacent warehouse buildings. In order to prevent unwanted communications between these two networks, all devices have a wireless network name configured in them.

Otherwise known as the ESSID (Extended Service Set I.D.), this is a basic method of only allowing certain wireless devices to associate with certain access points.

The following diagram shows how two overlapping AP zones only allow their own clients to transfer data.

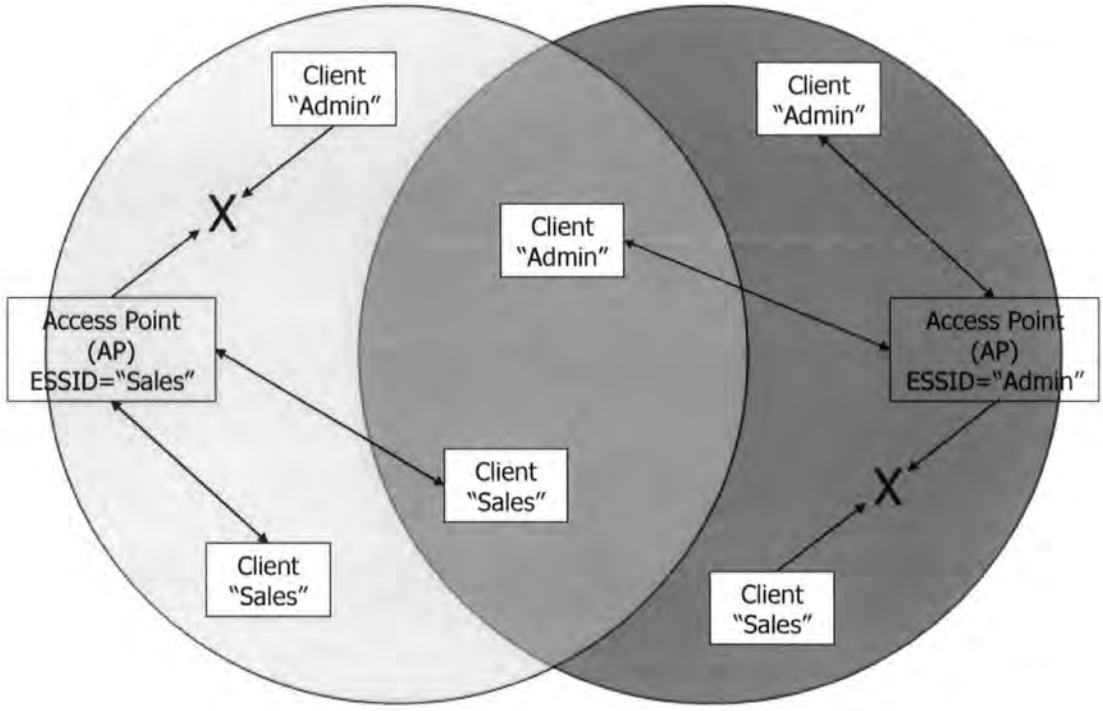

Figure 18: ESSID diagram

Clients will only associate with APs which have the same ESSID. Hence, the clients within the overlapped zone do not associate with the incorrect AP. Furthermore, the clients which are in the wrong AP area will not associate at all, and have no network connection.

2.3.7.2 Channel

If several APs are in range of each other, then the radio transmissions will interfere. To prevent this, there are different channels provided in the 802.11 framework. APs can

each be allocated a separate channel for communicating with the client devices. Wireless clients sequentially search on each channel, until they find an AP to associate with.

However, there are a limited number of channels available (13 in European countries[14]), and a large wireless system requires careful allocation of channels to minimise interference. The centre frequencies of the channels are 5MHz apart[15], but each channel broadcasts across a range of frequencies due to the nature of direct sequence spread spectrum encoding. This is illustrated below:

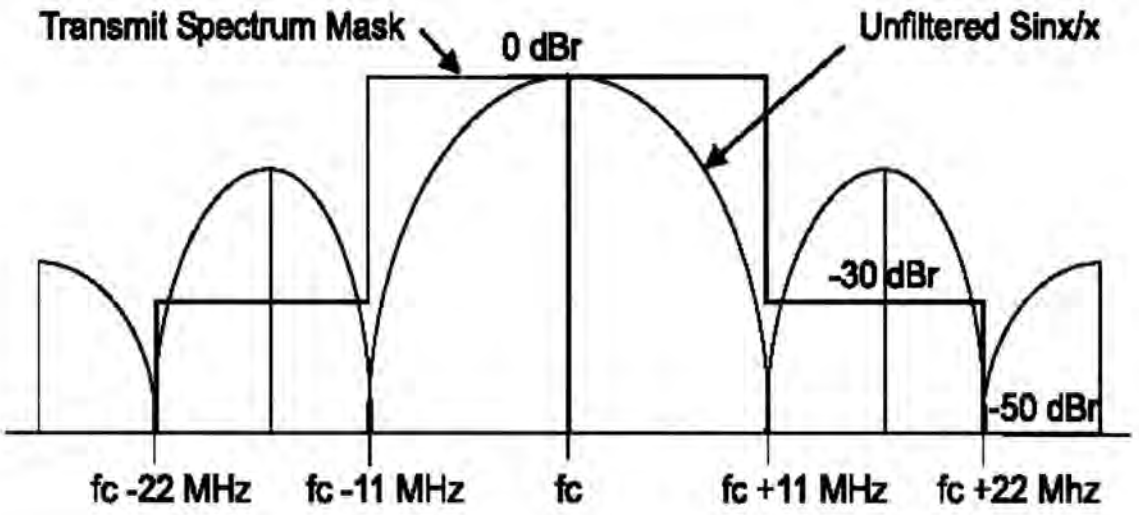

Figure 19: Spread spectrum transmission spectrum[16]

Because of the spread spectrum nature, any channels closer than 22MHz (or 25MHz when aliased to the nearest channel[17]) are able to interfere with each other.

2.3.7.3 Encryption Key

As shown in section 2.3.7.1, it is possible that wireless clients will be within range of another AP. Even if the ESSID is incorrect, it is still feasible that a client device could be programmed to intercept packets from a foreign network.

To prevent unauthorised network access, 802.11 allows for encryption. This is known as Wired Equivalent Privacy (WEP)[18].

For a client to associate with an AP, it must be using an accepted WEP encryption key on its data.

2.4 2.4GHz RF Electromagnetic Radiation

The particular carrier frequency used 802.11b transmissions in RFDTs is the 2.4GHz to 2.5GHz range; the Industrial, Scientific and Medical (ISM) band of the electromagnetic spectrum.

All electromagnetic (EM) radiation consists of an electric and a magnetic field[19] (E-field and B-field respectively.) These two fields oscillate sinusoidally in their respective planes, which are at right angles to each other. If these planes are defined as x and y in standard Cartesian terms, then the EM wave actually propagates in the z direction.

However, there are problems with using 2.4GHz RF for transmissions. These include:

- Limited transmission range due to 100mW output power[20] regulation;
- Numerous noise sources from other 2.4GHz users;
- Transmission and reflection by materials.

2.4.1 Output Power

This varies from region to region[21]. The USA permits 1000mW, i.e. ten times the permitted 100mW European limit. This means that 802.11b devices can range further from the AP in a USA system than in a European system.

2.4.2 Noise

The 2.4GHz frequency range used by 802.11b wireless Ethernet is an unlicensed band. This means that anyone is permitted to use it, providing that their devices do not radiate more than the allowed power.

As a formal definition, the band used is termed the "ISM" band – Industrial, Scientific and Medical. Common sources of noise include microwave ovens at 2.45GHz.

Large sources of noise, combined with a low output power, means that 802.11 signals are difficult to receive – the signal to noise ratio (SNR) is not good. However, the Direct Sequence Spread Spectrum (DSSS) radio copes well with poor SNR, when compared to other RF modulation. DSSS is described in a subsequent section of this thesis.

2.4.3 *Transmission and Reflection*

The high frequency RF transmissions are strongly affected by surrounding materials.

Some materials reflect microwaves with almost no transmission through them. Any metallic materials will have this effect.

Conversely, most other materials will absorb microwaves to a lesser or greater degree. As can be seen with microwave ovens, water and moisture greatly absorb microwaves (and hence increases in temperature when "microwaved").

These effects are significant in most warehouses or office buildings, and result in areas of the site not being covered by the access points.

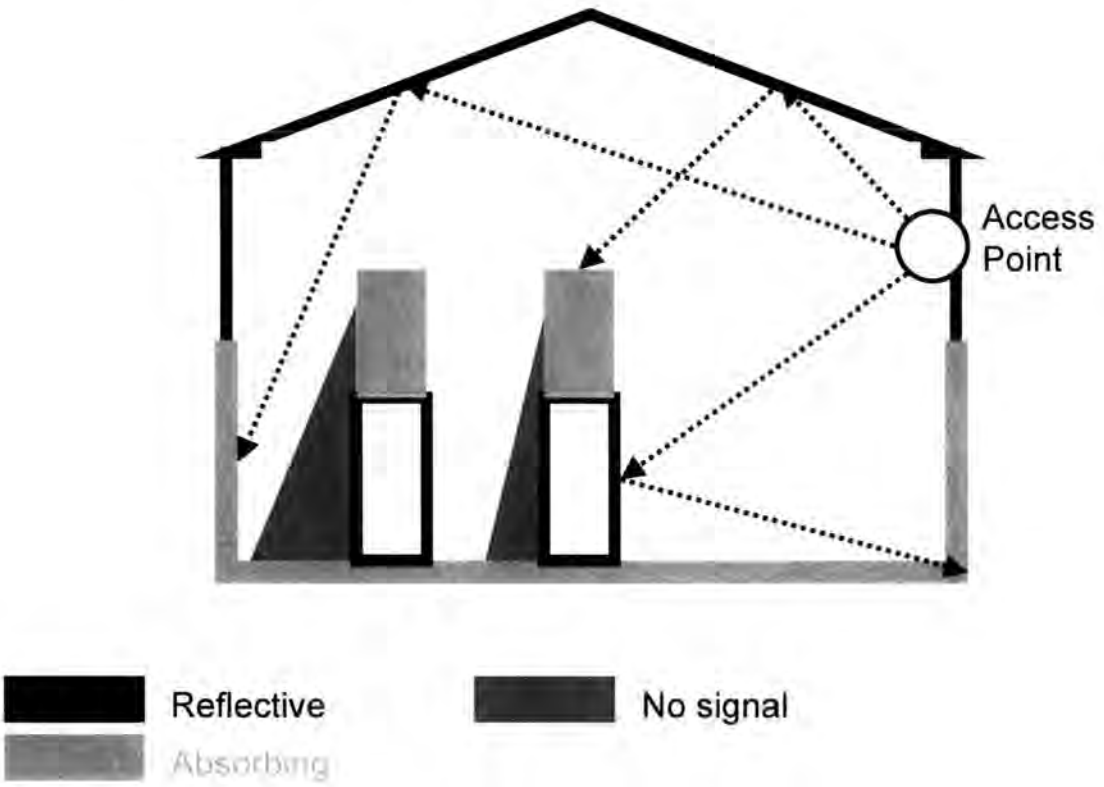

Figure 20: Idealised RF propagation around a warehouse

2.4.4 *Space Attenuation*

If radiation from the transmitter is completely isotropic (i.e. identical power in all directions), then received power falls off with distance due to the signal radiating outwards from the antenna.

2.4.5 Polarisation

The plane in which the electric field oscillates defines the polarisation of the EM wave. If, for example, the electric field plane is horizontal with respect to Earth, then the EM wave is said to be horizontally polarised. This has importance for antenna design, as the transmission and receiving antennae should be designed for EM radiation of the same polarisation. Antennae which transmit to each other should be polarised the same.

RF radiation can also be polarised circularly. In circular polarisation, the E-field plane is periodically rotating in time or space, with the B-field rotating in sympathy to retain the right angle between them.

2.4.6 Antenna Design

Antennae are designed for specific beam patterns and signal gains. A good antenna addresses the following points:

- 802.11 has a limited signal output of 100mW, so adding high-gain antennae means that weaker signals can be detected.
- Antennae can be directional, so that the 100mW output is directed entirely in the direction in which it is required. This may be all around (omnidirectional), or in a certain direction or plane only.

The gain patterns given below detail two different types of antenna.

The first, an omnidirectional antenna, is a general-purpose antenna which radiates RF power equally in all directions of the horizontal (azimuth) plane. It is good for general RFDT coverage.

The second pattern is for a highly directional "Yagi" type, which focuses RF power into a narrow beamwidth, and so is more useful for fixed point-to-point wireless network transmission; not for RFDT coverage.

2.4.6.1 Omnidirectional General Purpose Antenna

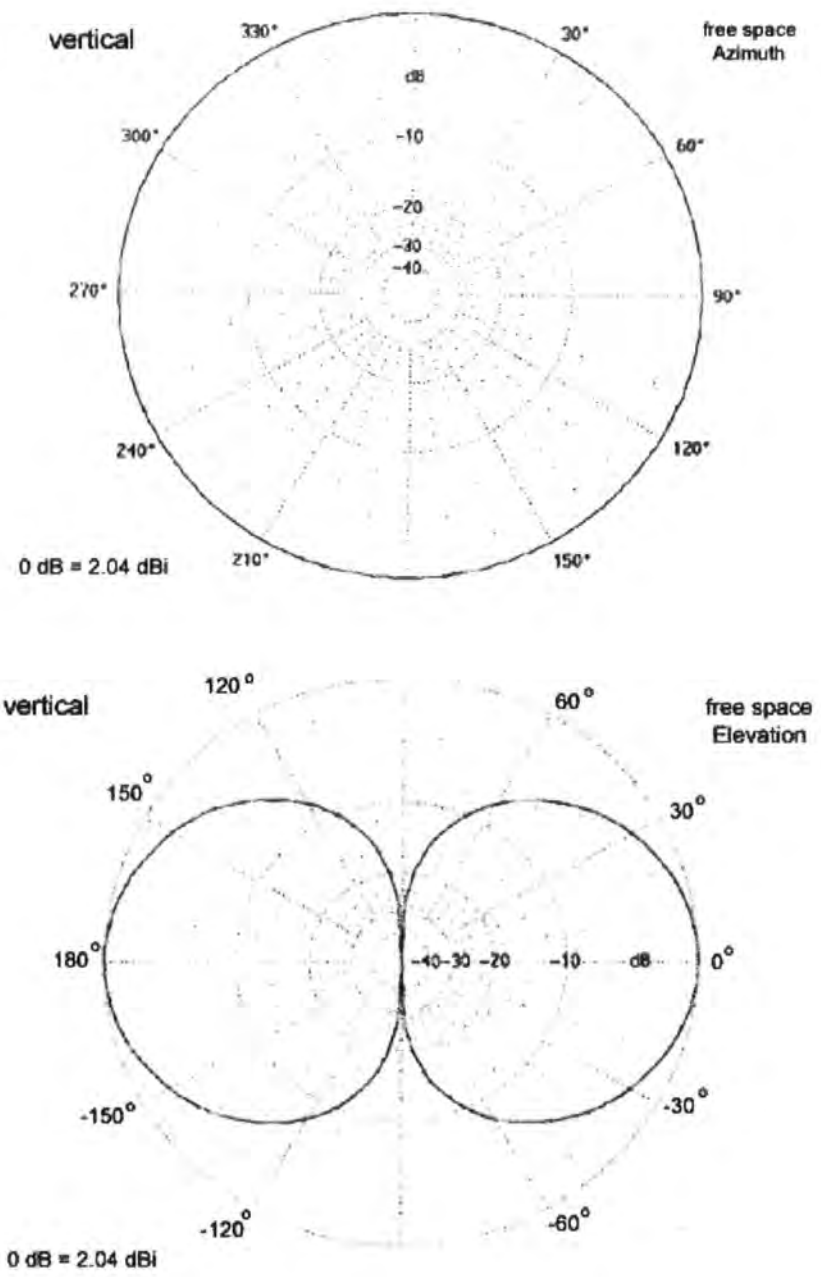

Figure 21: Omnidirectional antenna[22]

2.4.6.2 Directional Yagi Antenna

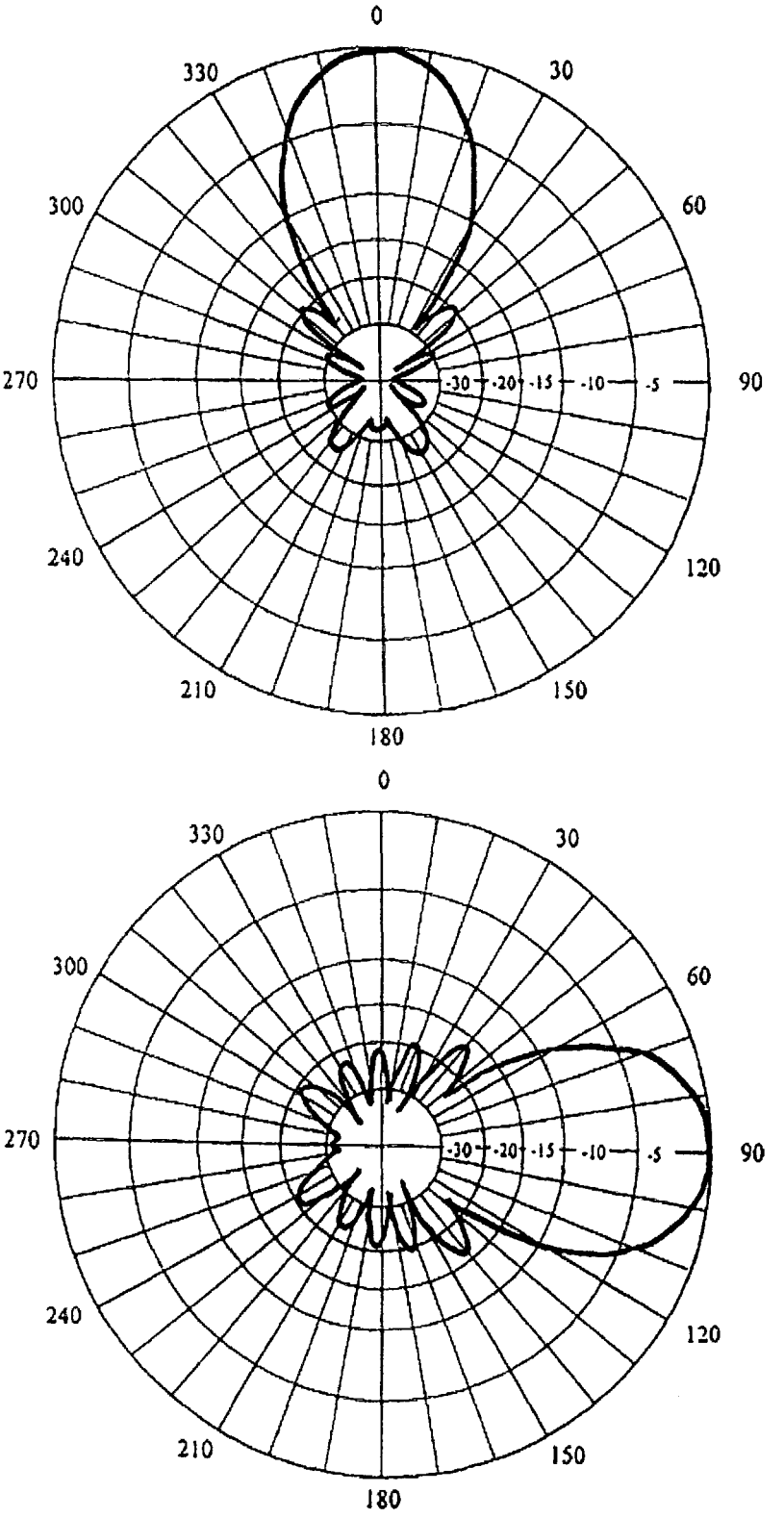

Figure 22: Directional antenna[23]

2.4.7 Installation

The problems outlined in the above section 2.4.3 mean that deciding where to install the access points is not straightforward, nor can it be decided without detailed knowledge of the materials and noise sources of a site.

In order to measure the radio propagation around a site, some experimental data must first be collected. This process is known as a site survey. By placing a test AP in a location, and then taking readings of signal strength around the site, a better suggestion of AP location can be made.

If a survey is not performed or the APs are mis-located, then full RF coverage may not be achieved.

This process of testing where to install APs is known as performing an RF Site Survey. It forms a major part of this thesis, and will be discussed later.

2.5 Spread Spectrum Communications

Most RFDTs communicate using the licence-free 2.4 – 2.5GHz frequency range, conforming to IEEE 802.11b. The 802.11b standard provides a means of modulating (i.e. encoding) data onto an RF "carrier" signal, of which there are 13 channels in Europe (see section 2.3.7.2)

The method of data modulation on the RF carrier is a technique known as Direct Sequence Spread Spectrum (DSSS), with Complementary Code Keying (CCK)

2.5.1 Advantages of Spread Spectrum

The following table[24] compares spread spectrum to some other methods of RF modulation.

	Source coding	Error coding	Encryption	RF modulation	Spread spectrum modulation
Data source optimisation	Yes				
Error detection		Yes			
Error correction		Yes			Yes
Tx security (not channel coding)			Yes		
Band allocation				Yes	Yes
Anti-eavesdropping			Yes		Yes
Tx authentication			Yes		Yes
Anti-interference					Yes
Anti-detection					Yes
Error reduction for given max. power					Yes
Anti-multipath					Yes

Table 6: Spread spectrum versus RF modulation

2.5.2 Outline of 802.11b CCK DSSS

An exact description of 802.11b DSSS encoding is beyond the scope of this section, but can be found in the IEEE 802.11b standards document[25].

A simple description of encoding the user data, in multiples of 8 bits, is as follows.

2.5.2.1 Fundamentals of CCK Codes

Central to the 802.11b DSSS implementation is CCK. This refers to a set of binary codes, 8 bits in length, which have special mathematical properties[26] in terms of their symmetry. 64 different CCK binary code sequences are used in the 802.11b 11Mbit/second implementation. Their symmetry makes it simple to distinguish between them, hence assisting in rejecting noise and interference.

2.5.2.2 CCK Codes within 802.11b 11Mbps

One way to consider how CCK encodes an input data byte, i.e. 8 bits, is as follows. There are 64 8-bit CCK codes available, and these can represent 6 of the input data bits (6 bits in binary is 64 combinations.) The remaining 2 input data bits are then used to differentially quadrature phase shift encode the entire output of the 8-bit CCK.

This is represented by the following method[27].

Firstly, the 8 input data bits d_0 to d_7 are split into 4 pairs. These pairs will encode four phase values, denoted as φ_1 to φ_4.

As described above, 6 of the data bits (i.e. 3 pairs, φ_2 to φ_4) form one of the 64 CCK codes, with each pair of bits encoding a φ phase value as follows:

Bits (d, d+1)	**φ phase value**
00	0
01	$\pi/2$
10	π
11	$3\pi/2$

Table 7: Phase values for CCK bit pairs, for φ_2 to φ_4

Next, the 2 remaining data bits encode the φ_1 phase value as follows:

Bits (d_0, d_1)	**Additional phase shift (even chip symbols)**	**Additional phase shift (odd chip symbols)**
00	0	π
01	$\pi/2$	$3\pi/2$
11	π	0
10	$3\pi/2$	$\pi/2$

Table 8: Differential phase value for for φ_1 term

Note that φ_1 is not solely based upon the first bit pair: it varies dependent upon whether the chip symbol number (see table below) is odd or even. This variation is by means of an additional 180 degree phase shift.

The 4 phase values are then substituted into the following formulae[28], to give the final chip symbol phase shifts, which are applied to the chosen 2.4GHz RF carrier frequency.

Chip Number	Formula
0	$e^{j(\varphi_1+\varphi_2+\varphi_3+\varphi_4)}$
1	$e^{j(\varphi_1+\varphi_3+\varphi_4)}$
2	$e^{j(\varphi_1+\varphi_2+\varphi_4)}$
3	$-e^{j(\varphi_1+\varphi_4)}$
4	$e^{j(\varphi_1+\varphi_2+\varphi_3)}$
5	$e^{j(\varphi_1+\varphi_3)}$
6	$-e^{j(\varphi_1+\varphi_2)}$
7	$e^{j\varphi_1}$

Figure 23: CCK code output chips (phase shifts) by input bit pair phases φ_1 to φ_4

Chip 0 is the first to shift the carrier phase, and does so for a time period equal to 11 million chips per second. Next, chip 1 becomes the carrier phase shift value. After that time has elapsed, chip 2 is used.

This continues until all 8 chips have been applied in turn, and then the CCK sequence is re-calculated for a further 8 data bits.

2.5.2.3 802.11b 5.5Mbps

802.11b defines that the data rate can drop to 5.5Mbps, to increase reliability: the 64 CCK code set is replaced by a 4 code (2 bit) set of more easily distinguishable codes.

2.6 Microsoft Windows Operating System

Windows PCs are often at the heart of a WMS. A brief review will be made of the functions which an Operating System (OS) should provide, with particular reference to Microsoft Windows NT4 based operating systems (NT4, 2000, XP)

2.6.1 General Operating System Tasks

An OS provides a code framework for the application to run in. This may include some or all of the following:

- Filesystem access and control
- Memory handling and allocation
- Input/Output functions (keyboard and screen)
- Application error and exception trapping
- Virtual memory (using the disc as sections of memory, to increase available memory)
- Multi-threading / Multitasking (executing several programs apparently simultaneously)
- Network handling, e.g. IP Sockets
- User interface, e.g. text command line, or Graphical User Interface
- Thin-client connections

The OS also provides hardware abstraction. That is, a range of different hardware may be supported, but via a common code interface for the software running on the OS: the software can operate across a range of different hardware.

2.6.2 Comparison of Operating Systems

	DOS	**Windows 95**	**Windows XP**	**UNIX / X GUI**	**Windows CE**
Filesystem	Basic ★	Medium ★★	Good ★★★	Good ★★★	Medium ★★
Memory	Basic ★	Medium ★★	Good ★★★	Good ★★★	Medium ★★
I/O & graphics	Basic ★	Medium ★★	Good ★★★	Good ★★★	Medium ★★
Exceptions	None	Basic ★	Good ★★★	Good ★★★	Good ★★★
Virtual Memory	None	Medium ★★	Good ★★★	Good ★★★	Basic ★
Multi-threading	None	Medium ★★	Good ★★★	Good ★★★	Medium ★★
Network Sockets	None	Basic ★	Medium ★★	Good ★★★	Basic ★
Windows	None	Good ★★★	Good ★★★	Good ★★★	Medium ★★
Built-in thin client	None	None	Medium ★★	Good ★★★	None

Table 9: Operating System features

2.6.3 Significant Features of Window OS

This thesis makes particular reference to the following features of the Windows operating system:

- Component Object Model (COM) interface, a standard for binary code libraries which allows cross-language calling of code
- Microsoft Foundation Classes (MFC) code library
- Threads and inter-thread messaging
- Windows and windows messaging
- Sockets and TCP/IP communications

2.6.3.1 COM Interface

Component Object Model (COM) is a Microsoft mechanism for providing cross-language compatibility in code libraries. For example, it allows a Visual Basic program to execute code written in C++. Many Microsoft components within Windows are implemented with COM interfaces, as is much third-party commercial code.

The following screenshot shows the Windows COM management tool, listing some of the COM interface servers installed on a particular machine:

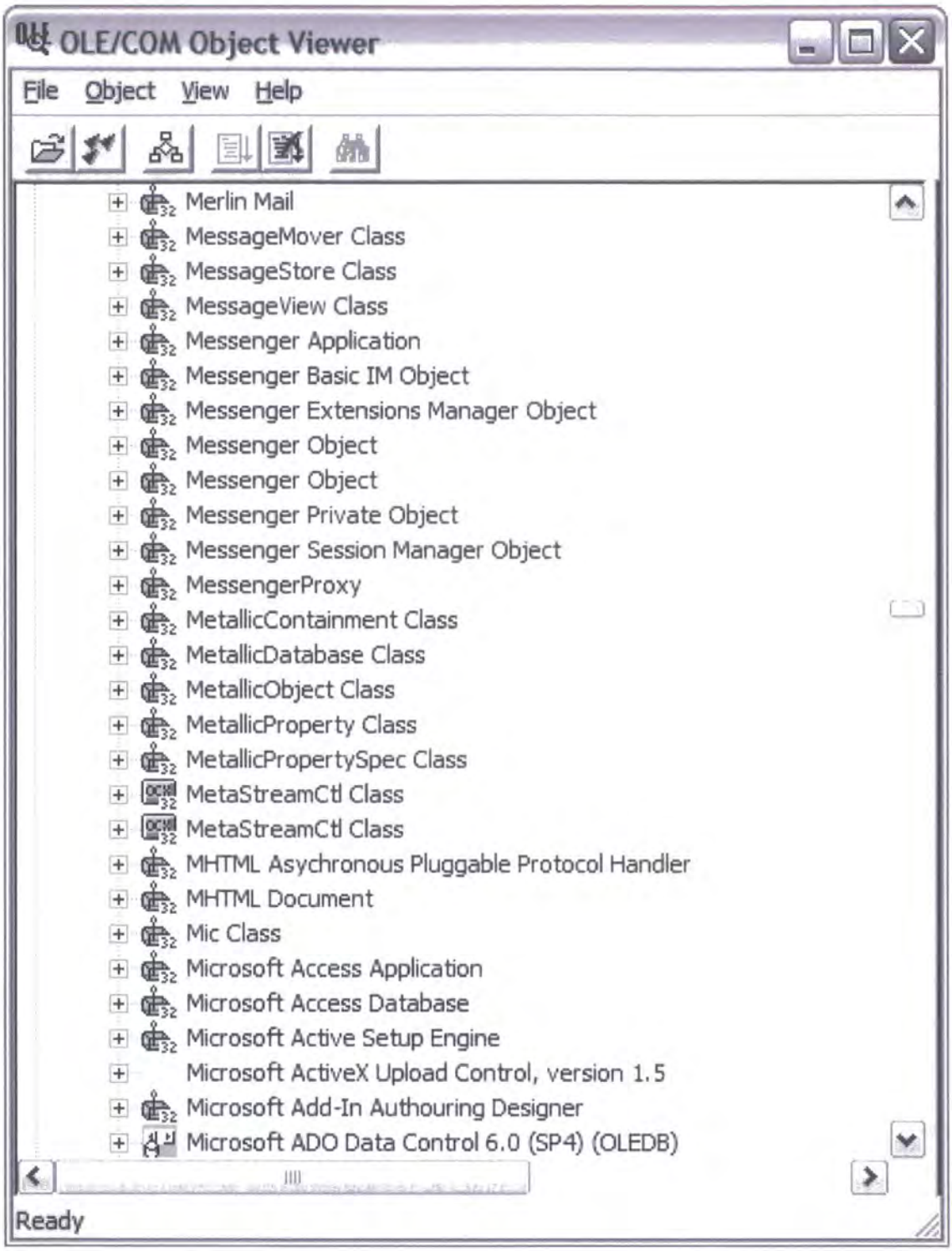

Figure 24: Spread spectrum block diagram

COM is the method of describing a code library. Each function has a memory pointer to its start address, and a description of how the functions pass data variables. The description of how the COM functions are accessed is described in IDL ("Interface Description Language"), which is compiled along with the library code:

Figure 25: COM IDL in Visual C++

Any COM-aware programming language can read the IDL for a code library, and access the code functions.

COM code libraries can be produced in two different types: in-process, and out of process. The former loads the COM code into the same memory area as main application which is calling the COM code functions; the latter separates the COM code into a different memory area and runs it as a separate application.

This thesis deals closely with COM, using it as the toolkit framework which can be called by other developers' Visual Basic programs. The type of COM used in this thesis is in-process, where the COM code is loaded into the host application's memory space, and COM calls are executed as a jump in program execution.

Microsoft's software development kits such as Visual C++, Visual Basic, and Visual Studio .net include a section called the "Object Browser". It allows the software developer to view what COM functions are available in each of the COM libraries on the computer. This includes COM functions provided by Windows; by other Microsoft applications; or by third party software tools.

2.6.3.2 Microsoft Foundation Classes (MFC) Code Library

The Microsoft Foundation Classes are a set of C++ classes which access many of the operating system functions. This includes filesystem, window control, threads, and sockets. Most applications written using Microsoft Visual C++ make heavy use of the MFC library, either directly or by inheriting from its classes.

Classes are particular to object orientated programming models. A "class" is a collection of functions which all operate upon the same data variables.

The most distinguishing feature of a class-based approach to programming is that a class can be "instantiated" many times, with each instance being an "object" and having its own independent set of variable data.

The common analogy for this is the "cookie cutter." One cutter (i.e. class) can be used to make as many different cookies (objects) as required, with each cookie having its own group of chocolate chips (variables).

In terms of the computer's memory space, this is highly efficient, because only one copy of the program code is stored (the class) regardless of how many times it is instantiated, and then the different groups of variables are stored separately for each object.

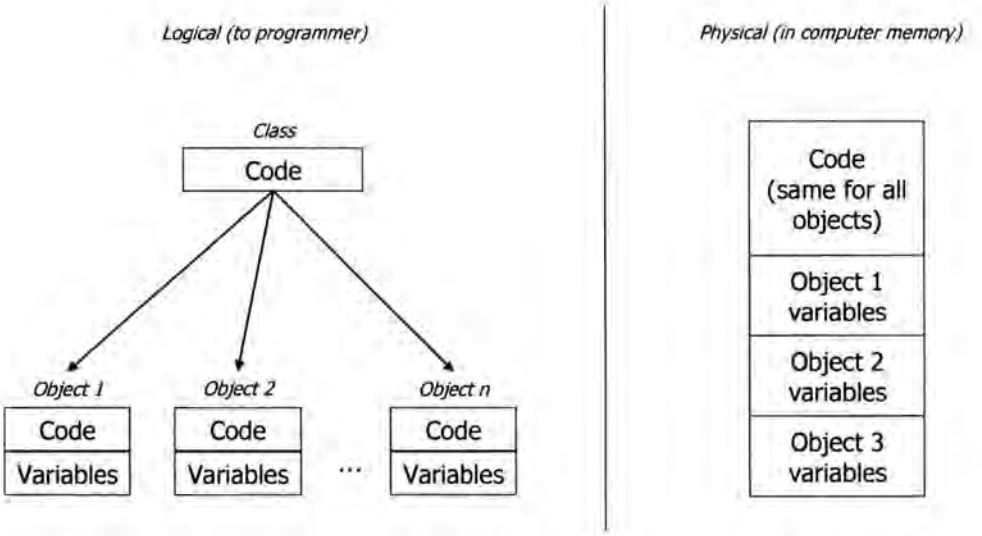

Figure 26: Classes and objects

The final advantage of a class based system is known as inheritance. A programmer can "inherit" their class from an existing one. In this process, all the existing classes' functions are automatically included in the inherited class.

The Microsoft Foundation Classes (MFCs), used heavily by C++ programmers on the Windows platform, are classes as defined in this section.

2.6.3.3 Threads and Inter-Thread Messaging

Windows has the ability to execute several program threads at once. This is actually performed by the OS rapidly switching program execution between the two threads, such that they appear to run simultaneously.

This is very useful for the programmer. For example, one thread could be written to handle the user interface, and another thread to listen to the network socket.

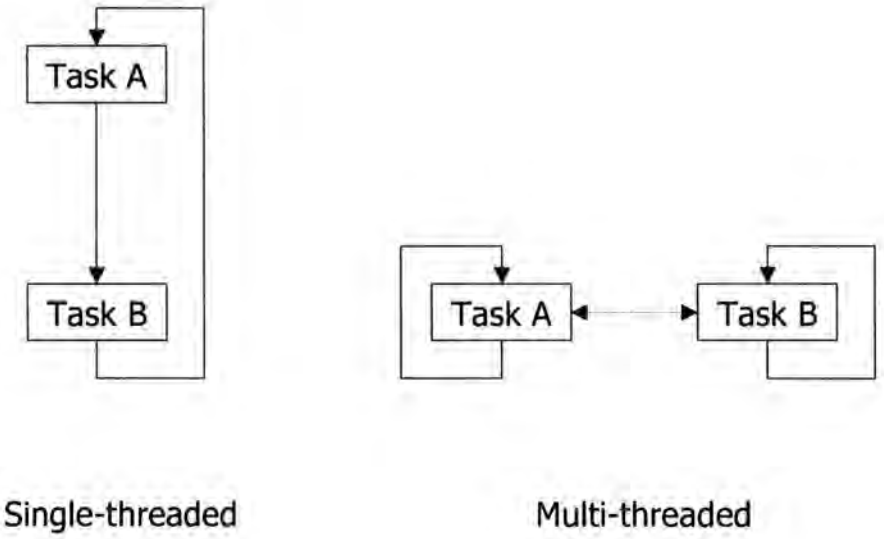

Figure 27: Single- and multi-threading

Without multi-threading, the programmer must specifically halt performing task A after a period of time, store its progress, and perform task B. After another period of time, task B must be halted, stored, and task A resumed.

This is complex and time-consuming for the programmer to develop.

With multi-threading, the operating system automatically alternates execution between threads, so that they appear to run simultaneously and in parallel. This method is called time-slicing.

The programmer can send inter-thread messages (ITMs) between them, which are placed in a First In First Out (FIFO) queue for each thread: upon each iteration around a central loop, the thread removes an inter-thread message from the queue and handles it.

The MFCs provide classes for multi-threading and inter-thread messaging.

A distinction should be made between threads and processes. Superficially, they are similar: they are both a method of executing code in parallel, provided by the OS. However, one process may have one or more threads, all of which are permitted by the OS to access the same memory area.

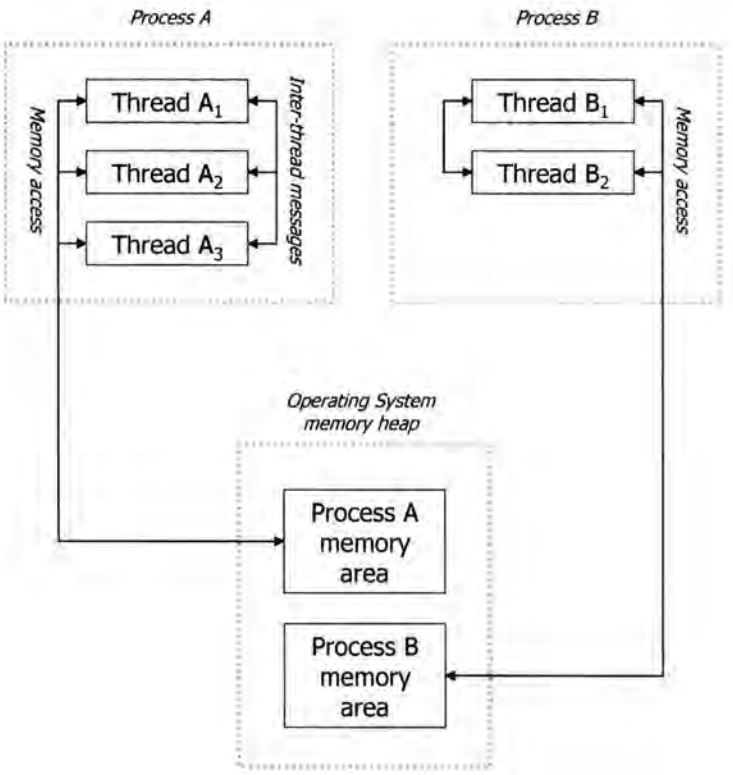

Figure 28: Process and thread memory areas

As shown in the above diagram, different processes occupy different memory areas, and the OS restricts processes to accessing their own individual memory areas. The OS then timeslices between each thread within each process, as represented by the following table (with reference to the above diagram)

Time	Code execution
t	Thread A_1
$t + x$	Thread B_1
$t + 2x$	Thread A_2
$t + 3x$	Thread B_2
$t + 4x$	Thread A_3
$t + 5x$	Thread B_1
$t + 6x$	Thread A_1
$t + 5x$	Thread B_2
$t + nx$	...

Table 10: Example timeslicing between threads

Note that the exact order and duration of the timeslicing is dependent upon many factors, and is controlled by the OS. It may not be as regular as the above table. However, the important point is that the variable x is a very small amount of time; hence, the threads appear to run simultaneously.

2.6.3.4 Windows and Window Messaging

Manipulation of windows is essential for providing an effective user interface. The Windows MFC library provides classes for manipulating windows, and the wide range of buttons and input fields which can be displayed on a window.

The Windows Messaging system is used for communicating between windows in an application. Code in one window class can place a "message" in the Windows message queue for another window class. Each window constantly runs around a code loop, which extracts messages from its FIFO (First In First Out) queue, and deals with them in turn.

Mouse clicks and key presses are two examples of Windows Messages. User-defined Windows Messages can be used in the code, to pass application-specific data between windows.

2.6.4 Microsoft Windows CE

When consumer PDAs (Personal Digital Assistants) started to become widespread, popularised by companies such as Psion Plc and Palm Computers, Microsoft developed a new OS for PDAs. This was termed Windows CE ("Consumer Equipment").

Many modern RFDTs use Windows CE as their OS, due to their PDA underpinnings.

CE was designed to emulate the feel of PC Windows, but designed from the ground-up for the limited hardware abilities of PDAs. For example, PDAs use a different Central Processor Unit (CPU) than x86 Windows PCs. The most popular CPU family for PDAs is the "ARM" instruction set: ARM is a British company which designs highly power-efficient CPUs.

Windows CE is a generic name for several specific operating systems, including[29] HandHeld PC (H/PC), Palm-size PC, and PocketPC (PPC).

	H/PC & HPC Pro	**HPC 2000**	**Palm-size PC**	**PPC 2000 & PPC 2002**	**CE .NET**
Windows CE basis	CE 2	CE 3	CE 2	CE 3	CE 4 .net
User input	Keyboard	Keyboard	Pen	Pen	Keyboard or pen
Screen size	480x240 or greater	480x240 or greater	240x320	240x320	Any or none
Supported	Obsolete	Deprecated	Obsolete	Yes	Yes

Table 11: Versions of Windows CE

2.6.5 Software Development Kits

Due to the complex nature of the Windows OS, software development kits (SDKs) are highly tailored for Windows so that they integrate with the various Windows code libraries.

This thesis focuses on the Microsoft Visual Studio 6 SDK, which was the current Microsoft SDK for Windows at the time of the project. This includes Microsoft Visual Basic 6, and Microsoft Visual C++ 6.

2.6.5.1 Visual Basic 6

Visual Basic (often referred to as VB6) is a popular language, due to its ease of programming. It has simple integration with COM components, and hence most Microsoft libraries can be used with ease. This includes Microsoft's Office applications, which can be easily controlled from VB. VB is excellent for implementing business-logic in applications.

The detriment of this ease of use is relatively slow execution speed, and a lack of access to very low-level functions. In this thesis, Castle Auto I.D. Solutions develop many applications using VB.

2.6.5.2 Visual C++ 6

Visual C++ 6 (or VC++) is Microsoft's implementation of ANSI C++, with links in to the Microsoft Foundation Classes (MFCs) for accessing most of Windows' libraries. VC++ is also the recommended language for developing COM components, due to its high execution speed and low memory overheads.

In this thesis, VC++ was selected for developing the toolkit, so that VB could be used for the business-logic of the applications (e.g. Warehouse Management Systems) with the VC++ code handling the lower level RFDT control.

2.6.5.3 Visual Studio .net

At the time of completing this thesis, Microsoft is replacing the Visual Studio 6 SDK with a new version: Microsoft Visual Studio .net (pronounced "dot net"). However, .net has only become available at the end of this project: any references to code libraries and development languages within Windows are with respect to Visual Studio 6, unless otherwise specified.

Visual Studio .net provides a new framework (".net") which replaces both MFC and COM. This framework is available from any of the .net development languages,

including C++, Visual Basic, and C# (Microsoft's own language based upon C++, but with more similarities to Java)

One positive aspect of .net is that both MFC and COM are still supported, albeit as legacy systems. This means that the toolkit developed in this thesis is still usable from .net.

The first release of Visual Studio .net was in 2002, under the name "Visual Studio .net 2002". The current release, at the time of writing, is "Visual Studio .net 2003".

3 RFDT Overview

Given that the requirements of the RFDT have been discussed (section 0), and that the background technologies have been discussed (section 2), the RFDTs themselves will now be discussed.

3.1 Purpose of RFDTs

As outlined in section 1.4.4.2, the RFDT is designed to speed up operations by entering data rapidly and accurately into a system. Their key features are:

- Portable, to allow data collection over a wide floor area
- Screen, to give feedback and instructions to the operator
- Barcode reader, to enter data rapidly and accurately
- Radio transceiver, to update data in real-time
- Keypad, to enter non-barcoded information

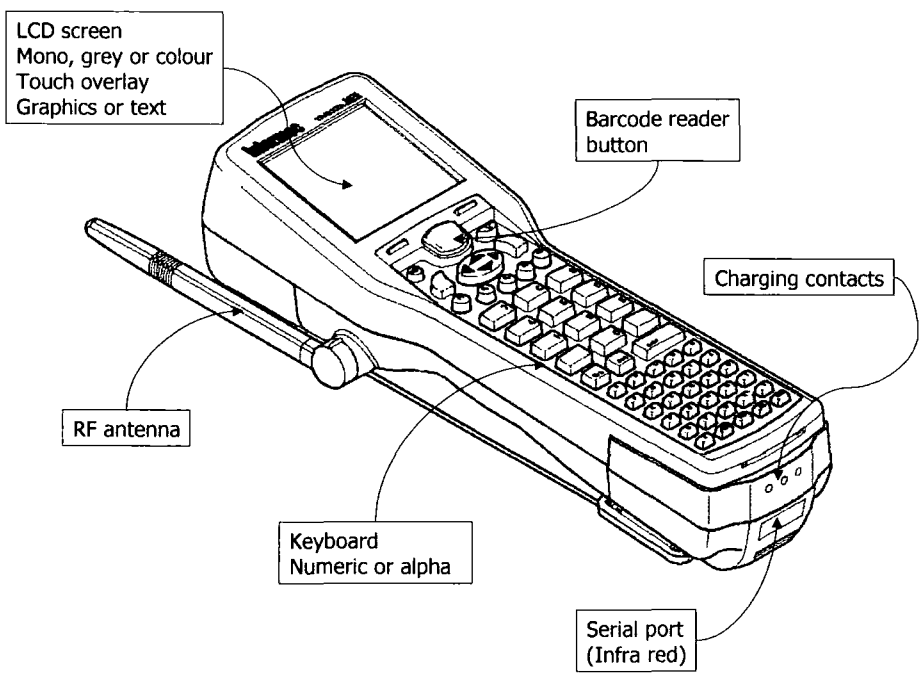

Figure 29: Illustration of a basic RFDT

3.2 System Configuration

The RFDT is just a component in an overall system. The following diagram shows how an RFDT might fit into a complete system.

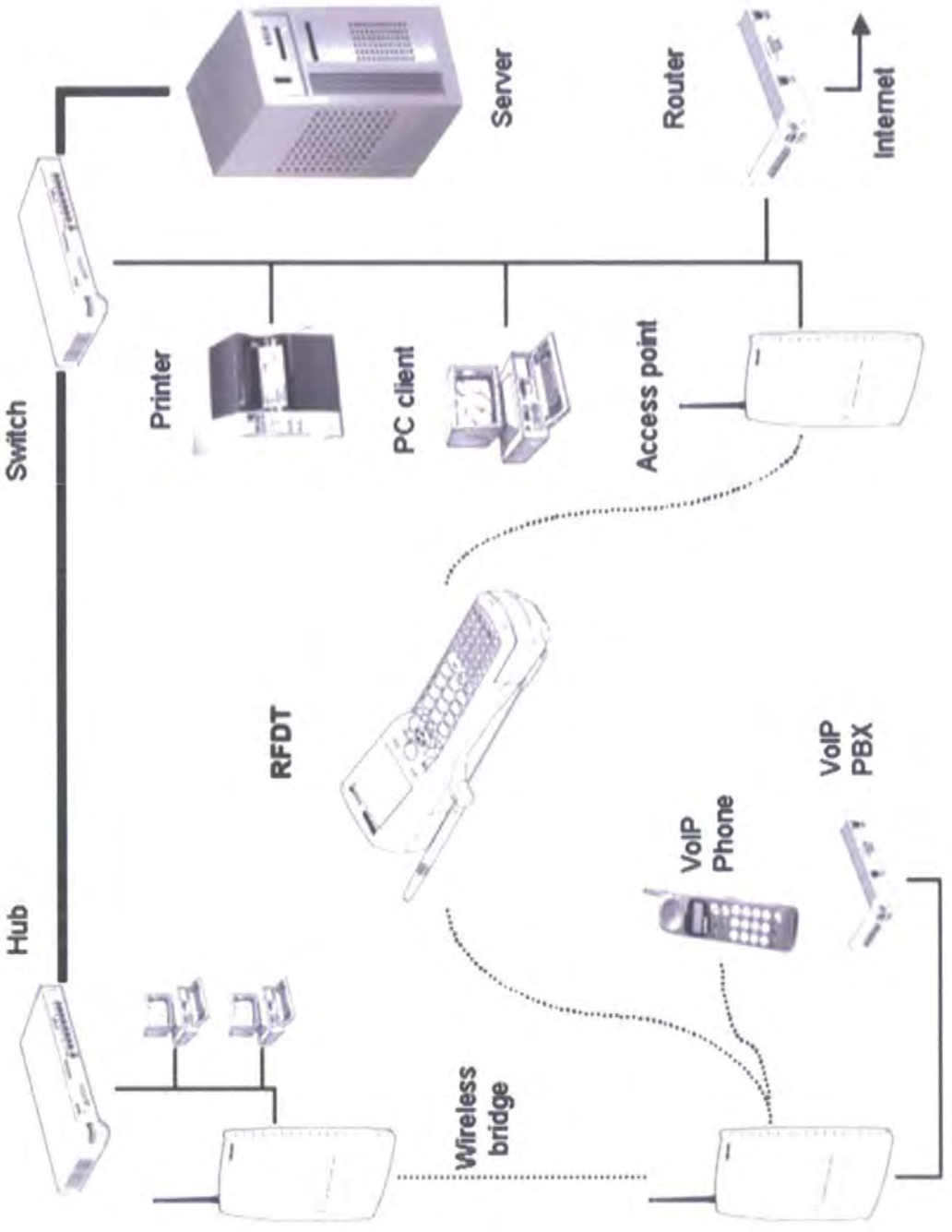

Figure 30: System overview

The system is typically coordinated by a server, which is the central store of information, and controls the flow of data in the system. It is important to note that

although the RFDT collects the data, it is the server which performs the majority of the processing on that data.

3.2.1 Network Connection

In order to transfer data to the rest of the network, the RFDT uses its wireless network card. This sends and receives data from the wireless Access Points. Standard TCP/IP networking is implemented. This was described in section 2.2.

3.3 Platform

In order to reduce development costs and standardise parts, RFDTs are usually based upon existing platforms which are used in the consumer market. Most RFDTs fall into one of the following:

- DOS based. Older RFDTs are often based upon 80x86 PC platforms, running a version of DOS;
- Windows CE based. Newer RFDTs are based upon Windows CE, and so can use any processor which Windows CE supports.

Below are comparison tables, showing how RFDTs are similar to consumer PCs or PDAs.

3.3.1 DOS Based RFDTs

	Intermec Trakker[30] 2425 DOS based RFDT	**Legacy desktop PC equivalent**
CPU	80386	80386
Operating system	DOS 6.22	DOS 6.22
RAM	640KB	2048KB
File store	2MB flash RAM	20MB hard disc
Graphical output	128x160 CGA	320x256 CGA or better
Textual output	20x16	80x25 or better
Network	TCP/IP over IEEE 802.11b	TCP/IP over Ethernet

Table 12: DOS RFDTs versus DOS PCs

3.3.2 Windows CE Based RFDTs

	Hand Held Products Dolphin[31] 7400 RFDT	**Hewlett Packard Jornada[32] 720 (consumer PDA)**
CPU	StrongARM SA-1110, 206MHz	StrongARM SA-1110, 206MHz
Operating system	Windows CE 3.0	Windows CE 3.0
RAM	32MB	32MB
File store	32MB Flash RAM	Battery backed RAM
Graphical output	256x320 grey	640x256 colour
Network	TCP/IP over IEEE 802.11b	Optional IEEE 802.11b
Drop-test	Survives 1.5m to concrete	n/a
Moisture/particle ingress	IP-64 rated	n/a

Table 13: WinCE RFDTs versus WinCE consumer PDAs

3.4 DOS Based Versus Windows

The older generation of RFDTs were often DOS-based, and designed for textual displays. However, newer generations are based on Windows style operating systems, (e.g. Windows CE[33]) and have graphical displays:

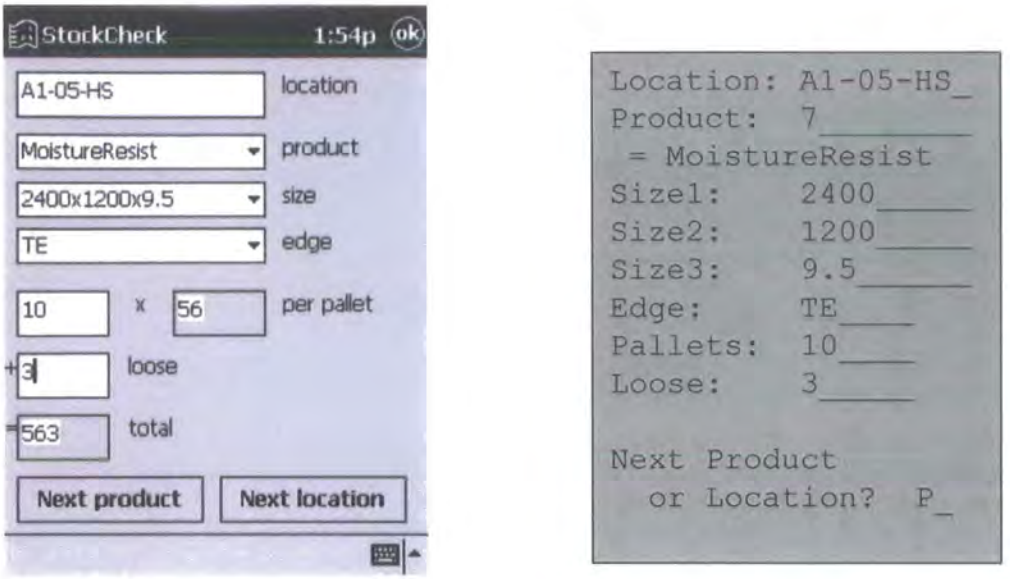

Figure 31: Graphical (left) and textual (right) display comparison

Each system has advantages and disadvantages, as shown in the following table.

	DOS-based	Windows interface
Cost	Cheaper ★★★	More expensive ★
Configuration	Simpler ★★★	More complex ★
Touch-screen interaction	None	Automatic ★★★
Graphical output	Limited ★	Extensive ★★★
Textual output	Good ★	Excellent ★★★
Multitasking	Limited ★	Automatic[34] ★★★
Cross compatibility	Poor ★	Good (common OS) ★★
Programming tools	Bespoke ★★	Similar to desktop PC[35] ★★★

Table 14: DOS based versus Windows based RFDTs

As can be seen from the above table, the Windows based terminals are more flexible and easier to develop software for. As such, the range of Windows terminals on the market is broadening. The cost of these devices is falling, and so they are replacing the DOS terminals. A more detailed examination follows.

3.4.1 Configuration

DOS terminals usually have customised OSs which are based upon a variant of DOS. They are set up specifically for running the RFDT, and require little more than configuring IP settings.

Windows terminals inherit some of the complexities of desktop PCs. This includes:

- Modifying the Registry settings for IP configurations;
- Locking the Taskbar so that the user cannot load other applications;
- Configuring the radio card driver

3.4.2 Touch-screen Interaction

Windows CE has built in touch screen support. Controls on the screen can be activated just by the user tapping them. The advantage is that applications can be made simpler for the operator:

- Drop-down list boxes for selecting from a pre-defined list of items, replacing the need for memorising product codes;
- OK/Cancel buttons, to avoid having to type "y" or "n" to questions;
- Touch screen operation is not supported on DOS devices.

3.4.3 Multi-tasking

DOS devices run one application at a time, and each application has one process thread, known as single-threading. However, Windows devices can run many applications at once, and each application may have numerous threads running in parallel: multi-threading.

Multi-threading enables the programmer to perform several tasks in parallel, with the OS automatically time-slicing between the tasks. A DOS application would have to perform these tasks in sequence, which can be difficult if one of the tasks does not return immediately.

3.4.4 Cross Compatibility

It is preferable for a system not to be restricted to one particular manufacturer or model of RFDT. For example, if a manufacturer ceases production of a particular model, then it is advantageous to introduce a replacement RFDT without the need for re-programming the system.

Software which is written specifically for a particular DOS RFDT is unlikely to compile for a different manufacturer's RFDT. This is because the code libraries are manufacturer specific. However, Windows CE uses the same basic Software Development Kit regardless of model, and so the same code can be compiled for a range of RFDTs.

The implications of cross compatibility will be discussed further in section 4.3

3.5 RF Barcode Readers

These are simpler devices than RFDTs, in that they are not derived from PC or Windows CE platforms, and do not use TCP/IP networking or Video Terminal (VT) screens.

RF barcode readers are simply a laser or CCD imager (see section 2.1.2) connected to a desktop PC. However, instead of the barcode reader connecting directly to the PC (via RS232 or keyboard PS2 port), the barcode reader sends data via narrowband RF to a basestation. The basestation then connects to the PC via RS232.

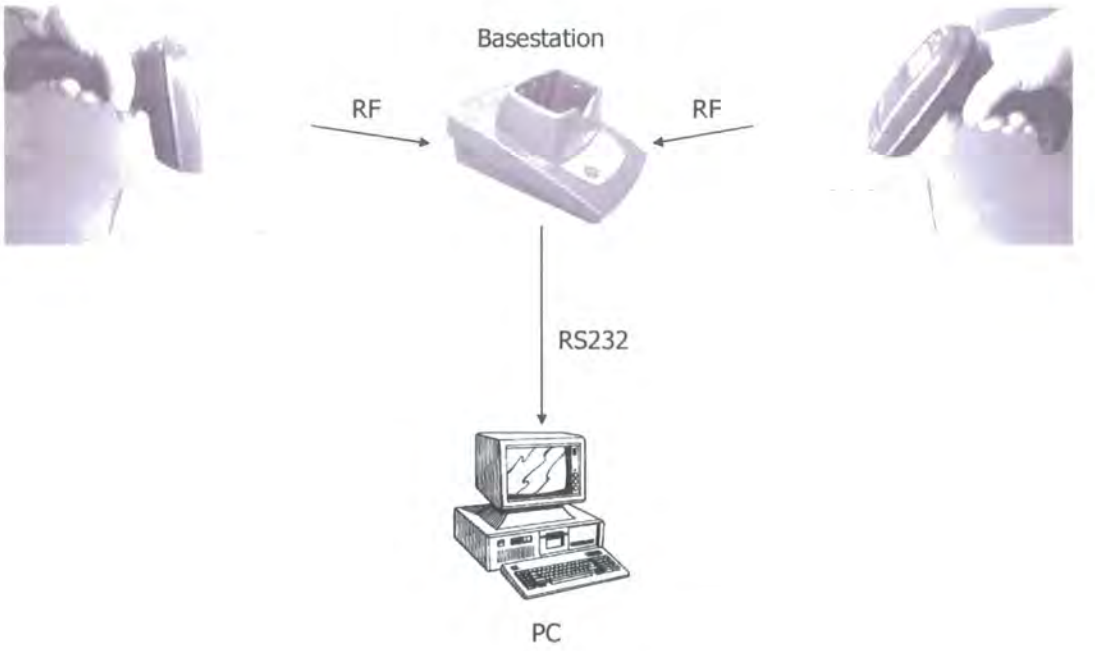

Figure 32: RF barcode readers and basestation

The added complexity over a normal barcode reader is the use of RS232. This requires the software to handle RS232 communications, which necessitates constantly checking the computer's RS232 port.

4 Software Toolkit Objectives

RFDTs are usually introduced, as outlined previously, into some kind of warehouse management system. The software which controls the system is often customised specifically for each user, and sold as a bespoke package. Performing a ground-up solution for each customer is time consuming, and therefore expensive.

The aim of this thesis was to identify the common components of such a system, and build up a toolkit to enable quicker software development.

4.1 The Toolkit Solution

Although most data collection systems are bespoke, there are some common operations which must be implemented on the RFDT. These will vary little from system to system, to a greater or lesser degree. Therefore the aim of this thesis is to identify the common operations, isolate them into re-usable code modules, and demonstrate how these can used to produce commercial bespoke systems.

These re-usable modules will be broadly termed as a software "toolkit". The toolkit should embrace all common operations which are performed on the RFDT, and also operate with as large a range of RFDT hardware as possible.

4.2 Common RFDT Operations

The following categories outline what project-independent operations an RFDT might perform:

- Display information on screen
- Send data back to server
- Read barcodes
- Process data internally

The software toolkit must be able to address certain key features in the above list, but more importantly, it must implement the features for a wide range of RFDTs. There are

several manufacturers and standards for RFDTs, so as large a number of terminals as possible must be catered for. This has implications for how the toolkit is written.

4.3 Standardising the Toolkit

The toolkit will be most effective if it can support the widest possible range of RFDTs, with the least possible model-specific coding.

It is not possible to cater for all RFDTs, and therefore this section shall examine which class of RFDTs to standardise upon.

As detailed in section 3.3, the platform has an effect upon how cross-compatible an RFDT is. The newer Windows CE devices use a common Software Development Kit (SDK), and this implies that Windows CE is a better option upon which to standardise.

However, this deduction is based upon the system being "thick client." This will now be discussed.

4.3.1 Thin Versus Thick Clients

This describes the "intelligence" of the software on the RFDT. Thin-client systems rely heavily on the server to control what appears on the RFDT display, and user input is sent directly to the server. Thick-client systems work in the opposite manner: the majority of data processing occurs in the RFDT, and it displays results back to the user. Data transactions only occur to the server when absolutely necessary.

The following diagram illustrates thin-client operation. Note that each time data is entered, it is sent to the server for processing.

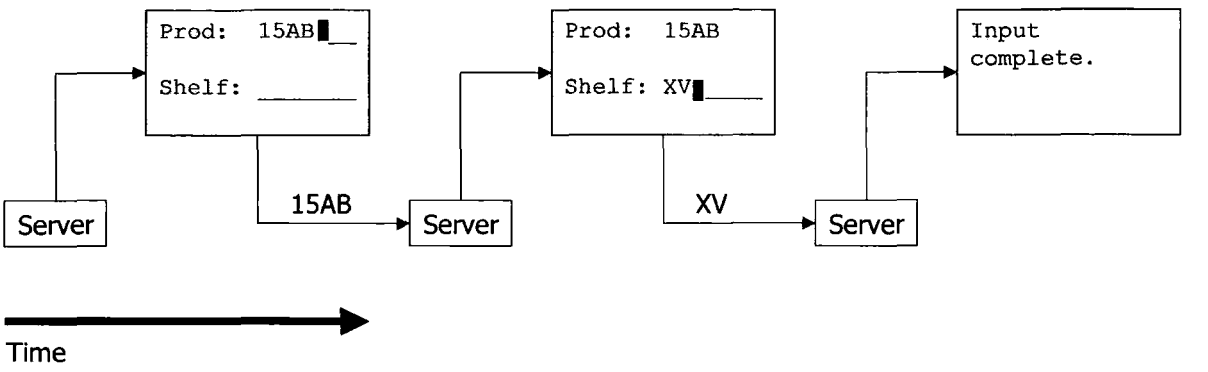

Figure 33: Thin client operation

Whereas the thick-client system sends data less frequently, as defined at program design time:

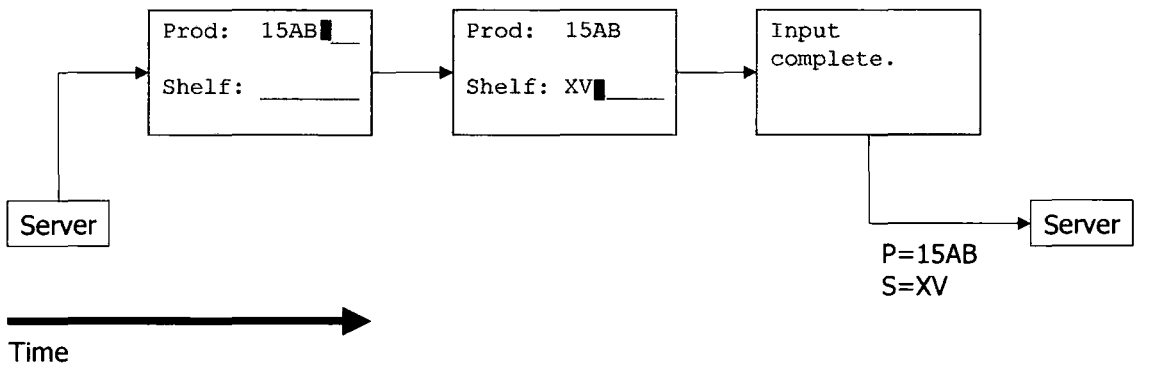

Figure 34: Thick client operation

The advantages and disadvantages of each system are as follows:

	Thin-client	**Thick-client**
RFDT software	Simple ★★★	Complex ★
Server software	Complex ★	Simple ★★★
RFDT cross compatibility	Excellent ★★★	Poor ★
Network traffic	Higher ★	Lower ★★★
Configuration	Server biased (simpler) ★★★	RFDT biased (complex) ★
Flexibility of display	Moderate ★★	Excellent ★★★

Table 15: Thick- versus Thin-client systems

4.3.2 Thick Client Description

With a thick client, custom software is developed specifically for the RFDT. A thick-client program can be as complex as any typical desktop PC program, making full use of the computer's facilities. A network of desktop PCs all communicating to a central database is an example of a thick-client system.

As discussed in section 3.3, the capability of the RFDT can vary greatly. Therefore a thick client system is tied strongly to the specific RFDT model: for DOS devices, cross-compatibility is poor; for Windows CE devices, cross-compatibility may only require a re-compile for different RFDTs.

Either way, it should be noted that cross-compatibility is restricted.

4.3.3 Thin Client Description

As described in section 4.3.1, a thin-client system relies heavily upon the server. User input is transmitted straight to the server, and the server then instructs the thin-client RFDT to update its display accordingly.

The thin-client system is used widely in the desktop computing world. Examples of desktop thin-clients include:

- Linux server with VT[36] Telnet dumb clients;
- IBM AS/400 server with IBM5250[37] dumb clients;
- UNIX server with X-Windows[38] clients;
- Citrix NFuse[39] Server with PC web browser clients.

These thin-client systems are compared in the following table:

	VT Telnet emulation	**IBM 5250 emulation**	**X-Windows emulation**	**Citrix NFuse web clients**
Basis	Text ★	Text ★	Graphical ★★★	Graphical ★★★
Server requirements	Simple ★★★	Medium ★★	Complex ★	Complex ★
Client requirements	Simple ★★★	Simple ★★★	Complex ★	Complex ★
Server config.	Medium ★★	Medium ★★	Complex ★	Complex ★
Client config.	Simple ★★★	Simple ★★★	Medium ★★	Simple ★★★
Cross compatibility	Excellent ★★★	Good ★★	Poor ★	Good ★★
Touch screen or mouse interaction	None	None	Excellent ★★★	Excellent ★★★
Network traffic	Medium ★★	Low ★★★	High ★	High ★
Support on RFDTs	Excellent ★★★	Good ★★	None	None

Table 16: Thin-client systems

4.3.4 Selecting a Standard

Now that the various client types have been defined, an assessment can be made as to which type to standardise upon. The following diagram summarises the various types which are available.

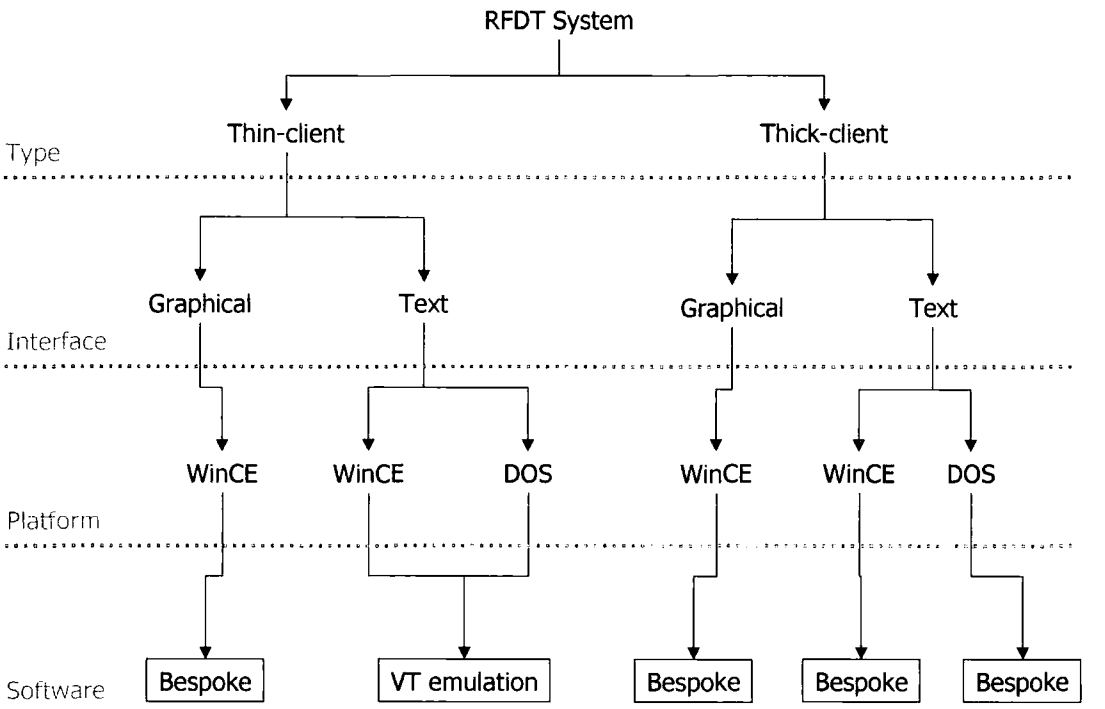

Figure 35: Summary of possible RFDT solutions

In the above diagram, each "bespoke" software solution is different for each path down the tree. Only the "VT emulation" solution can support all DOS and all Windows CE RFDTs, because VT emulation client software is so widely available.

Therefore if the toolkit is to operate with the widest possible range of RFDTs, it should be based around providing a VT Telnet solution.

4.4 Toolkit Objectives

The objective is to produce a software toolkit which will simplify the task of creating bespoke RFDT systems, with particular emphasis on VT, and operating with a broad range of RFDTs.

5 Toolkit Components

This section describes a list of requirements for the toolkit, and how they were implemented in terms of source code modules.

5.1 Requirements

The following is an analysis of what might be required for operating an RFDT based WMS. It is based upon previous bespoke systems installed by Castle Auto I.D. Solutions.

Broadly speaking, handling of the RFDTs should be split down into the following sections:

- Connection handling: TCP/IP is used, via the 802.11b radio card, to communicate between RFDT and server. The toolkit needs to facilitate this. Additionally, simpler RF barcode reader support should be considered, via the RS232 serial port (see section 3.5).
- User interface: A flexible user interface is required for the RFDT. Because the toolkit is using the VT thin client system, this means that code must be written to present an attractive VT formatted screen to the user.
- Data processing: Although quite specific to a particular application, e.g. a WMS, there may be some common operations for processing the inputted data. One example is database access.

5.1.1 Connection Handling

TCP/IP is used by the RFDTs, in common with most computer networking. TCP/IP is able to be routed across a Wide Area Network, and can be used by all current Windows PCs.

All devices using TCP/IP have an "address", and each device/address has multiple "ports" through which it communicates.

The following diagram illustrates how an RFDT (or any other client on a TCP/IP network) establishes a data connection with a server.

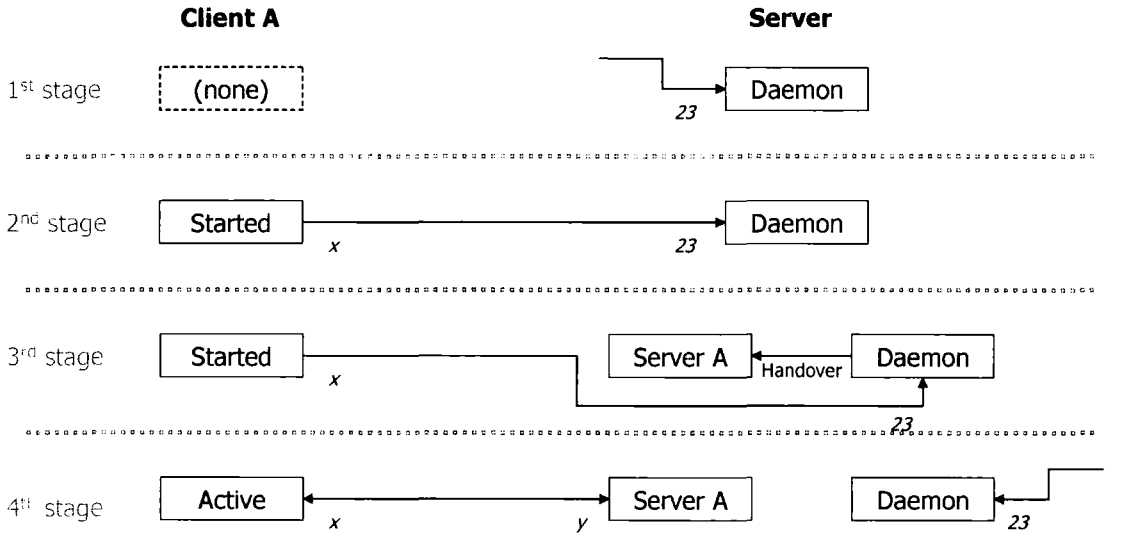

Figure 36: Establishing a TCP/IP connection to a server

In the 1st stage, the server runs a "Daemon" program. This continuously "listens" for any TCP/IP requests on a port. Specifically, in the case of VT Telnet, port 23 is dedicated as the daemon port.

In the 2nd stage, the client requests a connection from the server daemon. It uses an arbitrary free port from its pool (designated "*x*" in this diagram), but directs the request to the VT Telnet port (number 23 port).

The 3rd stage depicts the daemon program handing over the connection to a newly-created server program.

Finally, in the 4th stage, the client is dealing directly with the server, and the daemon returns to listening on port 23.

This form of connection handling should be included in the toolkit.

In addition, RF barcode readers using RS232 should also be supported.

5.1.2 User Interface

There are two aspects of the user interface, reflecting both sides of the application:

- VT Telnet user interface
- PC server user interface

5.1.2.1 VT Telnet User Interface

Video Terminal (VT) is a method for displaying text on a client computer. The server, or host, controls where the client cursor is on the screen, and then the client draws text in that position.

VT Telnet refers to the TCP/IP network port number 23.

The VT standard was originally a proprietary computer interface produced by DEC (Digital Equipment Corporation). As other companies began to produce equipment which was compatible with DEC's Video Terminal (VT) systems, so it became a widely adopted standard.

There are many different levels of VT compliance. A study of VT documentation[40] reveals the following major milestones:

Standard	Year	Comment
VT05	1970	72x20 upper case character display. Direct cursor positioning, with line erase ability.
VT52	1975	First set of Escape Codes implemented, to control the display (e.g. cursor positioning)
VT100	1978	80x24 upper or lower case display. "Block mode" support, where characters are only sent upon a CR.
VT220	1983	8 bit character set support. Other character sets in addition to ASCII, such as simple graphics fonts. Functions included to scroll sections of the screen up or down. User defined fonts, sent in the VT data stream.
VT320	1987	Vector graphics support.

Table 17: Development of the VT standard

Further developments were made, but beyond VT220 the features are not generally implemented in RFDT VT emulations.

The toolkit should provide an easy-to-use code interface for the following on-screen functions:

- Displaying text at a screen location
- Processing function key and cursor key commands
- Inputting text from a screen location
- Providing scrolling lists output
- Selecting different fonts

The nature of the code interface will be described later.

5.1.2.2 PC Server User Interface

The chosen language and development for this project was Microsoft Visual C++, on the 32bit Windows x86 platform. VC++, to use its shorthand, has a windows programming library called Microsoft Foundation Classes (MFC). MFC is a large number of C++ classes which encapsulate Windows functions.

However, there are still a number of common operations which the toolkit can provide. These include:

- Saving window positions when exiting the application
- Displaying a status line at the bottom of each window

5.1.3 Data Processing

Perhaps the most common data processing for a WMS is database access. Some form of code library should be provided in the toolkit for this.

Microsoft has a standard method of accessing various database types, called "ADO" (Advanced Data Objects). ADO has the advantage of abstracting the exact database format from the code.

As ADO is the currently recommended method of database access for Microsoft, some ADO code should be included in the toolkit.

5.2 Specification

Expanding upon the requirements, the following section describes the code interface.

5.2.1 Connection Handling

Two types of connection support is required: TCP/IP sockets (for RFDTs), and RS232 serial (for RF barcode readers)

5.2.1.1 TCP/IP Sockets

MFC already has good support for TCP/IP sockets, using its CAsyncSocket class. Toolkit code can be inherited from this class.

5.2.1.2 RS232 Serial

RS232 serial is not supported directly via the MFC library, so the toolkit requires a class for this. A significant factor is that RS232 access blocks execution on a thread, based upon a timeout. Therefore the toolkit RS232 handling should use a separate thread, to avoid the application hanging while reading the serial port.

5.2.2 VT Telnet User Interface

A C++ class should be provided to act as a text/input display box on a VT screen. This would have code for processing any input keystrokes, and creating a VT redraw string to redraw the contents.

In order to keep the code interface similar to a standard Windows textbox or listbox, the following class methods should be considered for implementation:

Method	Purpose
Xpos	Bottom left corner pos on VT screen
Ypos	Bottom left corner pos on VT screen
Width	Width of input/display field
Font	Set the display font (bold, invisible, etc)
UserInput	Process some user input from the VT Telnet connection, into the input field. Interpret as a complete line. If any control codes are sent (e.g. functions keys) then these should be interpreted also.
Redraw	Return a complete VT string which "redraws" the textbox on the VT screen. This will include positioning the cursor, outputting the text in the textbox, and setting the font.
AddListItem	If the VT field is a scrollable list, then items can be added using this method.
GetListItem	If the VT field is a scrollable list, get the item number which the VT user has scrolled to.
Callbacks	Callbacks to include: function key entered into the textbox, and new user data typed in.

Table 18: VT Code Interface

5.2.3 PC Server User Interface

It is important to make an application easy to use. This involves being careful about how the windows interface is designed. The toolkit should provide the following functions:

- Save the window positions and sizes when an application is closed. This means that when the application next starts, all the windows are in the user's preferred positions on screen.

- Re-size the controls within the window, when the window size is changed. Although Windows and MFC provide code for changing the boundary size of a window, the controls (buttons, text boxes, listboxes) within it do not change size. The toolkit should fix this: listboxes should expand to fill the window; "OK" buttons should be "pinned" to the bottom of the window, etc.

- Provide a window-width status bar at the bottom, into which helpful messages can be displayed.

A C++ class (termed "claWindowSize") to provide this functionality might have the following methods:

Method	**Purpose**
LoadWindowPositions	Load window and control positions from an .ini file.
SaveWindowPositions	Save window and control positions to an .ini file.
Init	Pass in a pointer to the parent window, which is to be resized, along with a minimum window size.
AddControl	Add a control from the parent window, so that its size and position can be controlled by claWindowSize. Flags whether the control should be resizable (x and/or y) or have its position pinned (top and/or left) should be included.
SetText	Set the status bar text.
OnWindowPosChanged	Parent control needs to pass any window re-size messages to claWindowSize, so that it can alter the control size/positions accordingly.

5.2.4 Data Processing

A base class will be provided to support the use of ADO database connections. The class, termed "claListObject", will encapsulate a database SQL (Standard Query Language) query into a class. This class will represent a record in a list, with the fields of the record easily available.

Although MFC provides a CADORecordBinding class for this purpose, there are many complexities involved with error trapping and status flags. These make the implementation quite complex.

The toolkit shall provide its own base class, "claListObject", which hides those complexities inside the following functions:

Method	Purpose
OpenSQLRst	Perform an SQL query on the database, and bind the resulting records to this claListObject.
GetFieldAsString	Get the field value as a string (e.g. for outputting to a listbox)
GetFieldName	Get the field name or description as a string (e.g. for the headings of columns in a listbox)
TotalFields	Get the total number of fields in the record.
IsEmpty	Are there no records returned from the SQL query.
IsBOF	Is this record the first one in the set.
IsEOF	Is this record the last one in the set.
MoveFirst	Move to the first record in the set.
MoveNext	Move to the next record in the set.
MovePrevious	Move to the previous record in the set.
MoveLast	Move to the last record in the set.
CreateNew	Create a new, blank record.
Delete	Delete this record.

6 Commercial Use of the Toolkit

The toolkit was put to use in several applications, over a period of two years. Experience of each application was fed back into the development of the toolkit, so that it expanded as it was used.

Its use in these commercial applications demonstrates how there is overlap between the applications, hence demonstrating how the toolkit approach has saved on development time.

6.1 List of Applications

The following applications were developed at Castle Auto I.D. Solutions, using the toolkit.

i. Dual scanner to dual printer link

ii. Weighscale to printer link

iii. Scanner / Printer / Mainframe link

iv. VT COM Server for Warehouse Management System

v. RF site survey system – NetTest/ProSurvey

These will each be described briefly.

6.1.1 Dual Scanner to Printer Link – BHD Windows Ltd

A bespoke system for BHD Windows Ltd, this software supports two RF laser barcode scanners, each one printing to its own printer. Products read with scanner A would output labels on printer A, and scanner B to printer B:

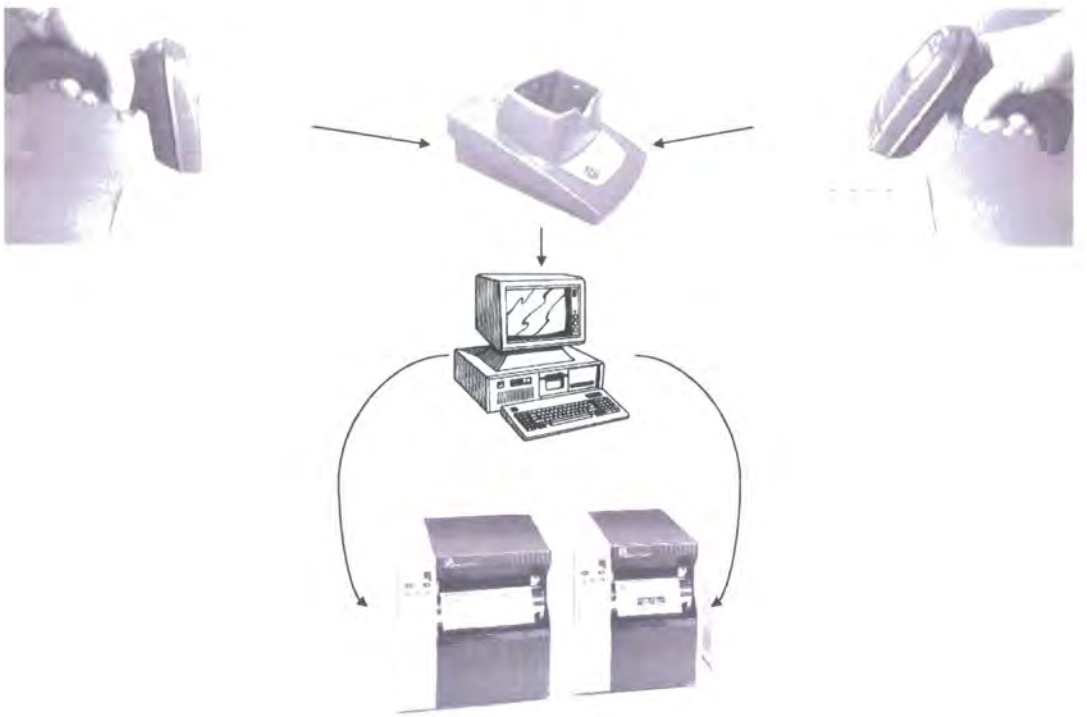

Figure 37: Dual Scanner to Printer Link

The scanner basestation (top centre) sends the scanned code to the PC's serial port, along with an identification code for which scanner the data came from. The toolkit software on the PC then instructs the label output software to send to the appropriate label printer.

6.1.2 Weighscale to Printer Link – Kezie Ltd

This system was required to interpret the output from a digital weighscale, and output a label containing the weight. The previous application (dual scanner to printer link) is essentially very similar: serial port input, routed to a label printer. The following diagram illustrates this.

Figure 38: Weighscale to Printer Link

6.1.3 Scanner / Printer / Mainframe Link – Black & Decker

Black & Decker Ltd European Repair Facility needed a system to scan repaired products, output a batch packing label, and enter the data into a 3270 mainframe computer.

This again has much similarity to the previous weighscale and dual scanner applications (sections 6.1.1 and 6.1.2), with added code to output results to the 3270 mainframe:

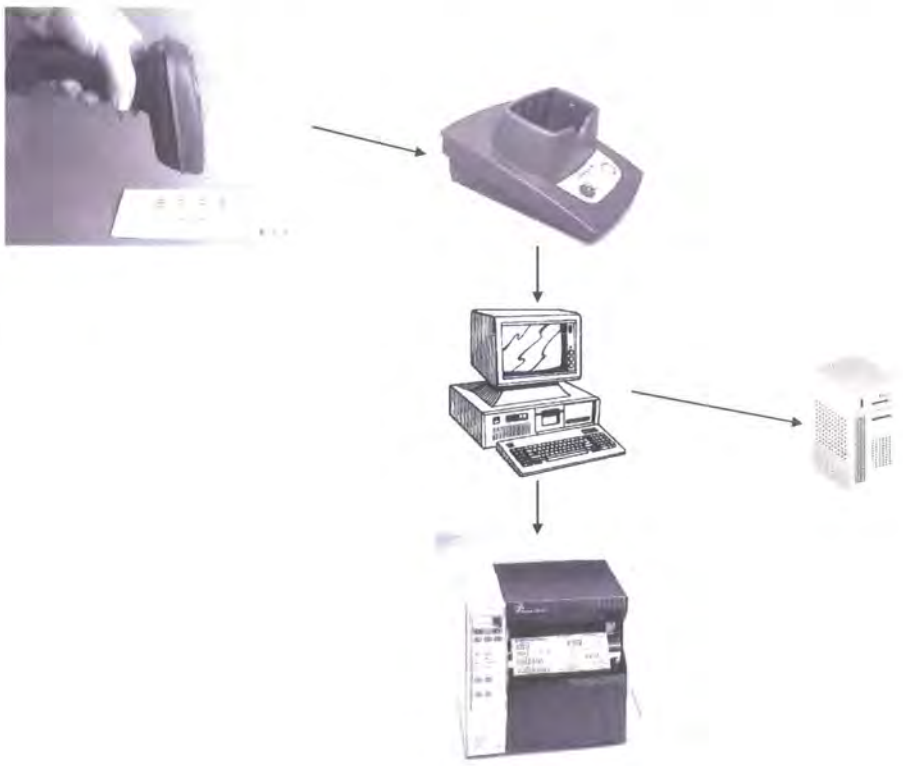

Figure 39: Scanner / Printer / Mainframe Link

6.1.4 VT COM Server – GeoLogistics Ltd

Designed for GeoLogistics Ltd, a warehousing and distribution company, this is a typical Warehouse Management System. The bulk of the program was written in Visual Basic, as speed of operation was not an issue (maximum of 6 RFDT connections.) However, the screen output and keyboard input from the RFDTs, using VT, was produced using the toolkit.

The toolkit was used to produce a Microsoft COM Server which implemented the VT screens. This allowed the Visual Basic developer to treat the RFDT display and input as if it were a Visual Basic control on a Windows Form.

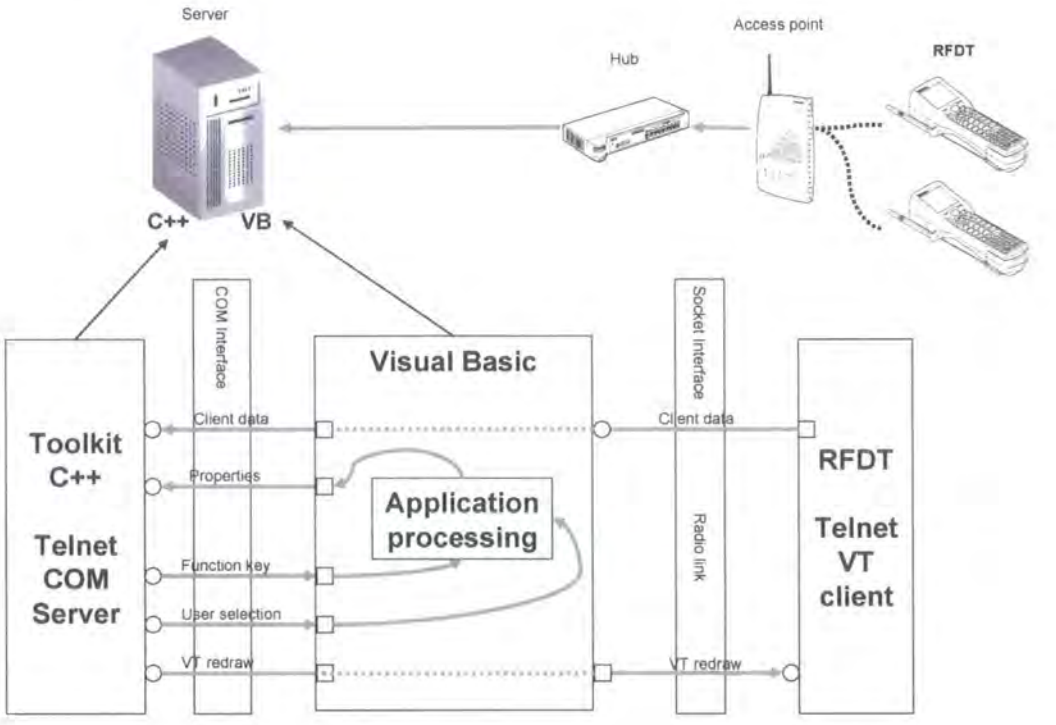

Figure 40: VT COM Server in WMS

The above diagram shows how the VB application transparently sends incoming VT data to the toolkit Telnet COM Server, and passes the redraw commands straight back to the VT client. The Visual Basic developer therefore has a simple programming interface for controlling the client screen via VT.

This is studied in greater detail in section 7.

An example of the VT screen output, using the toolkit VT COM Server, is given below.

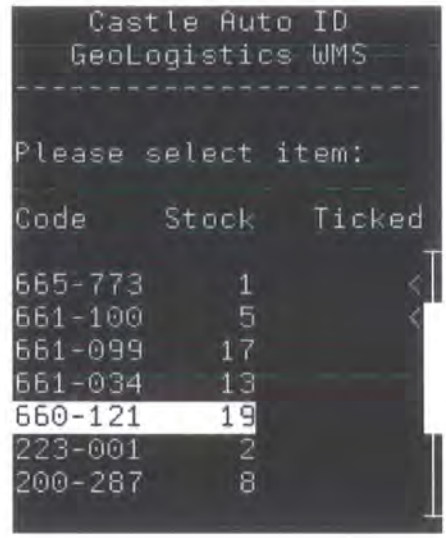

Figure 41: RFDT screen handled by the VT COM Server

Each field on the screen is a separate instance of the VT COM Server. This emulates a standard Visual Basic control, where a separate text box exists for each item. The lower half of the screen is a single VT COM Server instance, operating as a scrolling list.

6.1.5 RF Site Survey System - ProSurvey

Before installing the wireless access points for RF clients to use, it is necessary to perform a survey. This establishes where to place the wireless access points for best coverage (see section 2.4.7), and helps to achieve coverage over the desired area.

The established industry method for surveying is to walk around the site with the RFDT client, and observe a link test program. When the RFDT link test program reported that signal had been lost, the surveyor would mark the boundary on a map where that access point signal had ceased.

The author of this thesis decided to implement a more scientific solution, based upon recording samples of the signal strength at regular intervals across a grid.

The RF site survey system started as a small program making use of the VT COM Server code, so that a PC could be controlled via the RFDT client, with the PC recording the signal strength of the RFDT.

Figure 42: Early site survey screen handled by the VT COM Server code

Despite beginning as a small program for internal use, the success of this scientific surveying resulted in Castle Auto I.D. Solutions developing it into an off-the-shelf software package, called ProSurvey.

This application of the toolkit will be examined in more detail in section 8.

6.2 Overall Code Re-use

One objective of the toolkit was to re-use code between applications, hence reducing development time. The following diagram represents the amount of code used (size of ellipse) and amount re-used (overlap between ellipses) for the 5 applications listed in section 6.1.

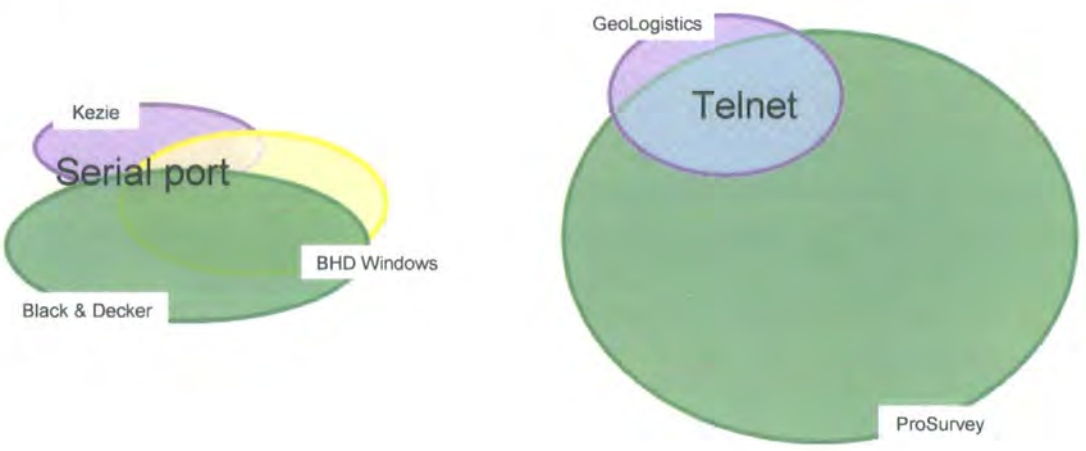

Figure 43: Toolkit code re-use for commercial applications

6.3 Examining Code Re-use

As an example of how code has been re-used at the C++ class level, the common code functions between the Kezie (weighscale to printer) and BHD Windows (dual scanner to dual printer) applications can be examined.

The diagram contains screenshots of the Microsoft Visual C++ development environment, showing the applications in "class view" – that is, the various C++ classes making up the applications.

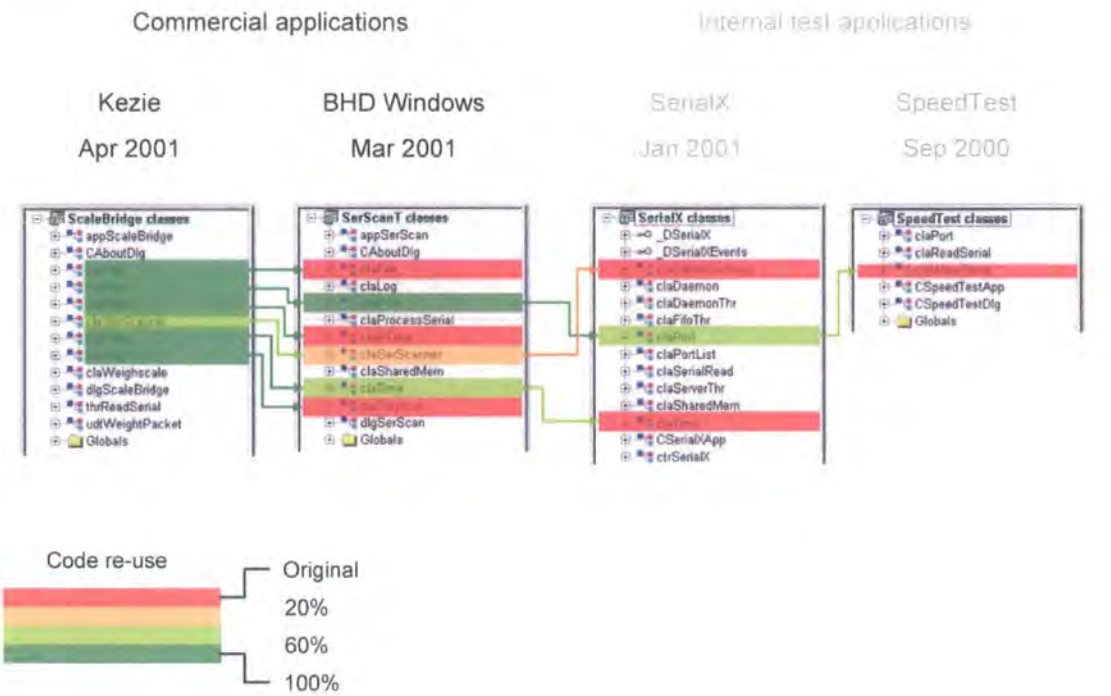

Figure 44: Toolkit code re-use example

The diagram traces some of the classes back to early test applications, developed for internal use. Detailed above, the Kezie (weighscale to printer) application has 5 of 12 classes 100% the same as BHD Windows (dual scanner to dual printer). One other class is approximately 60% the same.

The two commercial applications' code re-use is further highlighted in the table below.

Class	Code re-use from previous application (approx %)	
	Weighscale to printer (Kezie) c.f. BHD Windows	**Dual scanner / dual printer (BHD Windows) c.f. test applications**
claPort	100	100
claProcessSerial	Unused	20
claSerScanner	60	20
claSharedMem	Unused	100
claTime	100	60
claFile	100	0
claPTime	100	0
claTrayIcon	100	0

Table 19: Toolkit code re-use example

7 Case Study: VT COM Server

A commercial project at Castle Auto I.D. Solutions was proposed to use a database maintained by a Visual Basic application, which in turn would handle VT Telnet connections from several RFDTs.

In order to abstract the RFDT code from the main VB application, it was decided to provide VT support via separate code – specifically, code produced using the RF software toolkit described in this thesis.

7.1 Introduction

To interface between VB and the C++ RF toolkit, the Microsoft COM Server interface was chosen. A Microsoft COM Server is a binary code library which is cross-language compatible. That is, a COM Server written in Microsoft Visual C++ can be accessed from Visual Basic. This was described in section 2.6.3.1

COM Servers can be either in-process, running in the host application's memory space; or they can be out of process, having a separate memory space to the application. This COM Server is in-process, so the VB application jumps execution to the C++ compiled code during a COM Server function call.

The VT COM Server for this project was designed to perform the function of handling the VT data, and passing results back to the Visual Basic application.

This is illustrated in the following diagram.

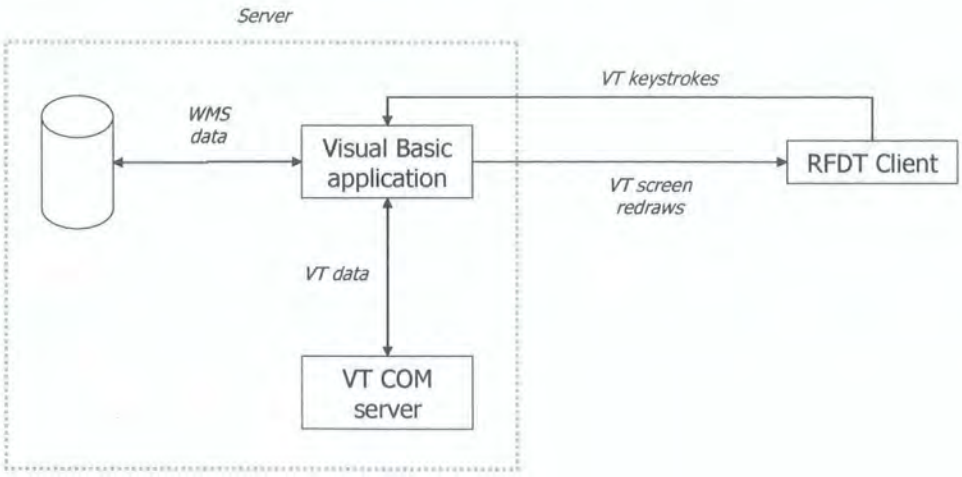

Figure 45: Telnet COM Server

The incoming VT keystrokes are not processed by the VB application, but are instead sent to the Telnet COM Server for processing there. The COM Server then returns VT screen redraw commands, which the VB application sends back to the VT Telnet client.

7.2 Requirements

Required user interface functionality for the VT interface:

- Interpreting inputted keystrokes in line or character mode
- Displaying single line text fields
- Displaying scrolling lists, which the VT client can scroll up and down.
- Different display fonts: bold, normal, "password" (typed chars are not displayed or appear as asterisks)

7.3 Design

As outlined earlier, a COM interface was chosen to interface the toolkit code to the VB application. Using the COM model to represent one text field per COM instance, the programming model would be very similar to a standard Visual Basic control.

In this particular case, the TCP/IP socket connection was implemented in the VB application. However, all data taken from and sent to the socket connection is handled by the COM Server.

This is illustrated below.

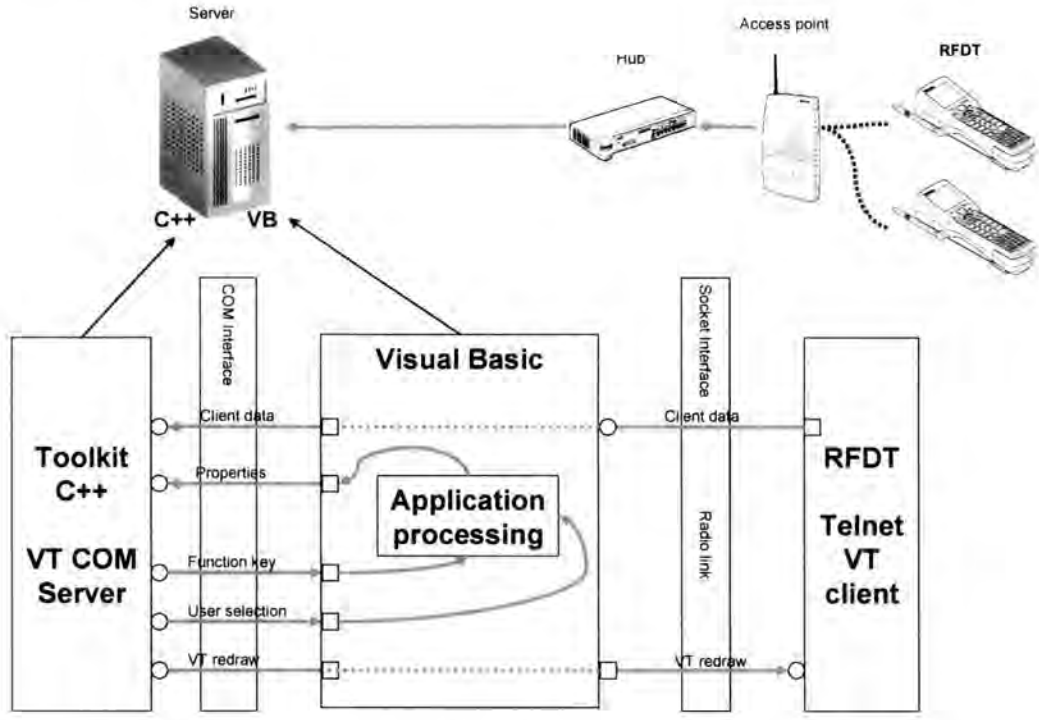

Figure 46: VT COM Server in WMS

7.4 *Implementation*

The COM interface can be used through Visual Basic, included as a standard COM type library. The name chosen for the type library is "VTILib" – a shortened form of "VT Interface Library".

A screenshot of the Visual Basic Software Development Kit is shown below, with the VT COM Server library ("VTILib") displayed in the VB Object Browser (a window displaying a COM Server's code interface)

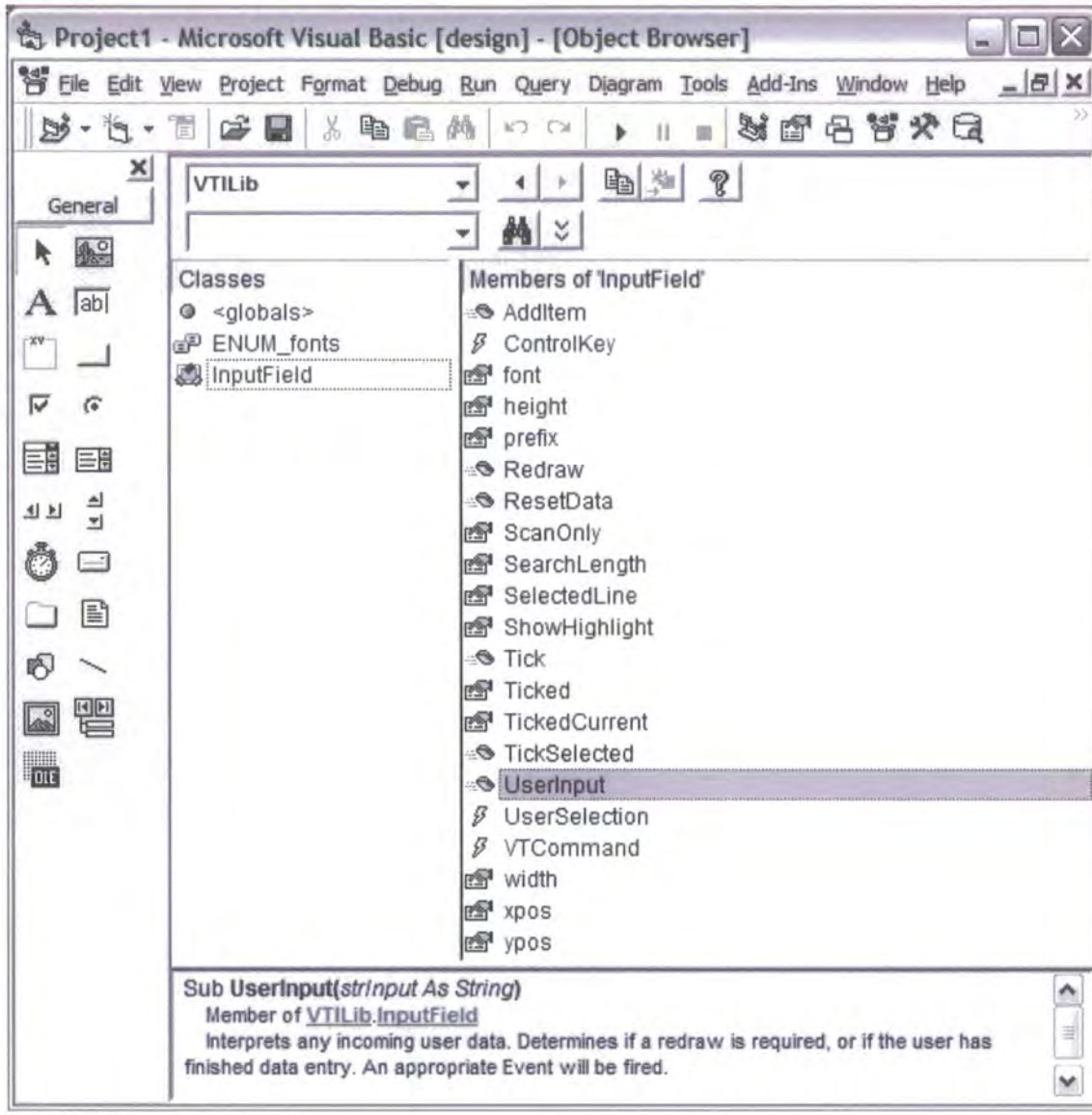

Figure 47: VT COM Server being used in Visual Basic 6

As shown in the Visual Basic Object Browser in the above figure, the VT COM Server provides assistance to the VB software developer by annotating the COM interface functions. A description appears in the bottom pane of the screen, describing what the selected function does.

The table below is a list of these functions and help strings.

COM interface function	**Purpose**	**Type**
xpos	x position of field on the VT screen.	Property
ypos	y position of field on the VT screen.	Property
width	Width of field on the VT screen.	Property
height	Height of field on the VT screen.	Property
ShowHighlight	Set to 0 for no highlight, or other for a highlight (default)	Property
font	Text effect to apply. See Font List.	Property
SearchLength	Partial search length, for user typing a value in a listbox.	Property
Ticked	Return True if the item is ticked in the listbox.	Property (read)
TickedCurrent	Return True if the currently selected item in the listbox is ticked.	Property (read)
SelectedLine	Return the value of the currently selected item in the listbox.	Property (read)
ResetData	Call this before creating a new input field.	Method
AddItem	Add a data item to a list box. Unnecessary for a normal input/display field.	Method
UserInput	Interprets any incoming user data. Determines if a redraw is required, or if the user has finished data entry. An appropriate Event will be fired.	Method
Redraw	Call this to force a redraw. This will fire a VTCommand event.	Method
Tick	Tick an item in the listbox.	Method

TickSelected	Tick the currently selected item in the listbox.	Method
VTCommand	VTCommand event. Fired to provide the host VB app with a VT redraw command, whenever a redraw is required (e.g. in response to a user input)	Event
UserSelection	UserSelection event. Fired to inform the host that the user has finished their input for this field, i.e. pressed Return.	Event
ControlKey	Event indicating that the user has pressed a control key (i.e. a function key, or cursor key)	Event

Table 20: VT COM Server code interface

7.5 Usage

The following screenshot shows how the VT COM Server is used by Visual Basic on one of the VT screens. Many instances of the VT COM Server are run simultaneously, with each instance handling its own field.

Figure 48: VT COM Server annotated in WMS screen

Some of the features of the VT COM Server are illustrated above. The listbox dominates the screen, with automatically implemented scrollbar along the right. The highlight bar is moved up and down by the user pressing the VT up/down cursor keys. Ticked items are shown in a brighter font, with a small "<" angle bracket to their right.

7.6 Implications of .net

As described in section 2.6.5.3, during the completion of this thesis Microsoft released a new software development kit: Visual Studio .net. The focus has been moved away from COM; however, due to the widespread use of COM, it is still supported as a legacy system.

The following screenshot shows the VT COM Server ("VTILib") being used in Visual Studio C++ .net, displayed in the object browser.

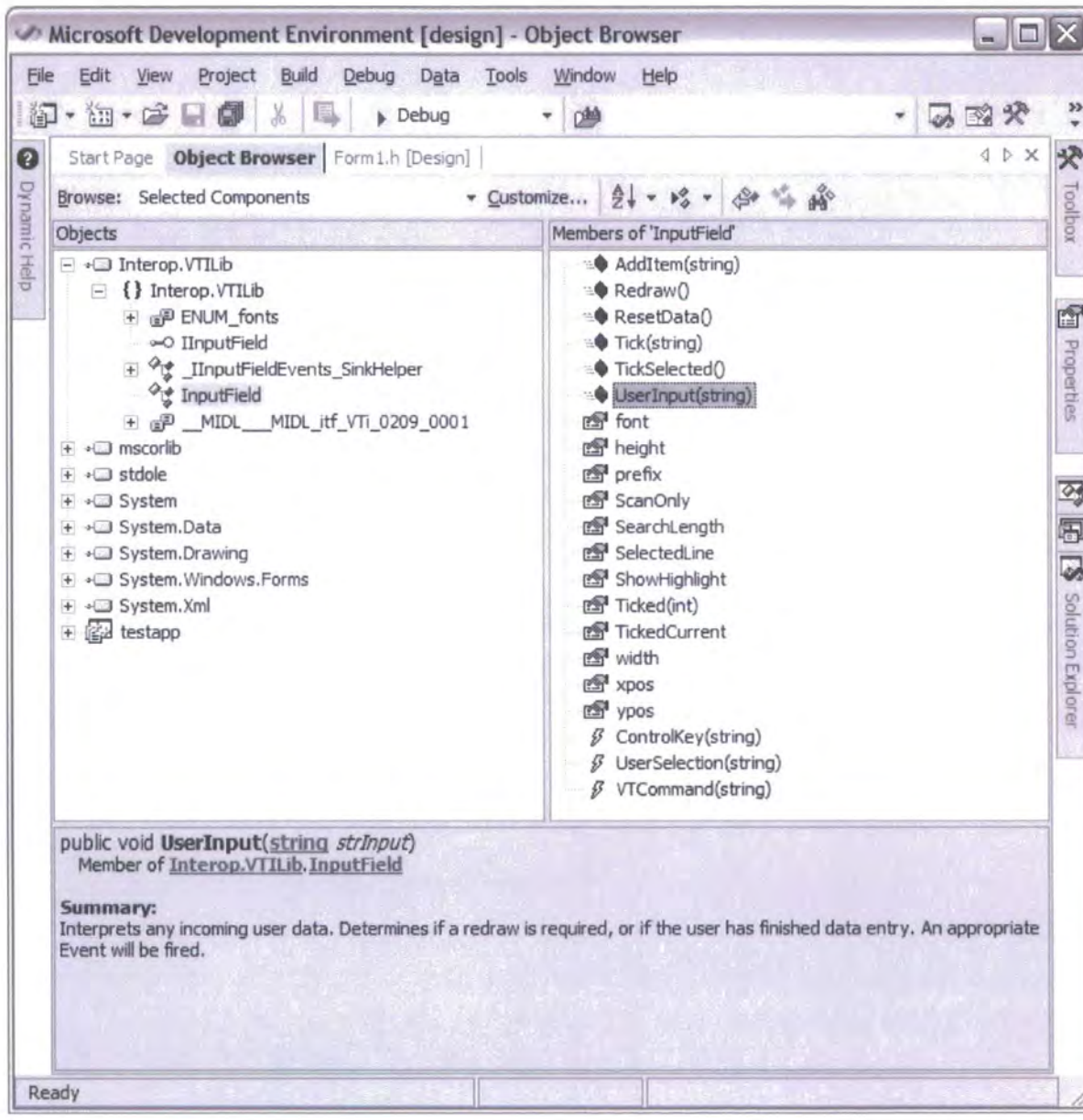

Figure 49: VT COM Server being used in Visual Studio .net 2003

8 Case Study: Site Survey System

One of the most important facets of an RFDT system is the 802.11b radio backbone. As detailed in section 2.4.7, experimental data must be taken at the installation site, to ensure that adequate radio coverage is given.

The prevailing method in the RF industry is to check for radio coverage around an access point, and establish the boundary where an access point runs out of coverage. This method pays no attention to quality of signal, or to measured RF performance.

The author proposed that Castle Auto I.D. Solutions should take a more scientific approach, by recording signal strength statistics at regular grid intervals across a site.

In order to facilitate this process, the site survey software was created.

8.1 Introduction

8.1.1 Naming

At the start of its development, the site survey software was known as "Network Test" and then "NetTest". Towards the end of its development, Castle Auto I.D. decided to sell the software as a commercial application. At this point, the software was given the more marketable name of "ProSurvey."

8.1.2 Survey Method

The principle of the survey software is to take signal strength readings (available from the IEEE 802.11b radio hardware) at many grid locations over the site. These readings can then be plotted to form a radio coverage map.

The following diagram shows where the signal readings ("samples") were recorded on a site, and how the resulting coverage map appears. Blue areas indicate good signal strength (typically near to the access points), and the red areas represent bad coverage.

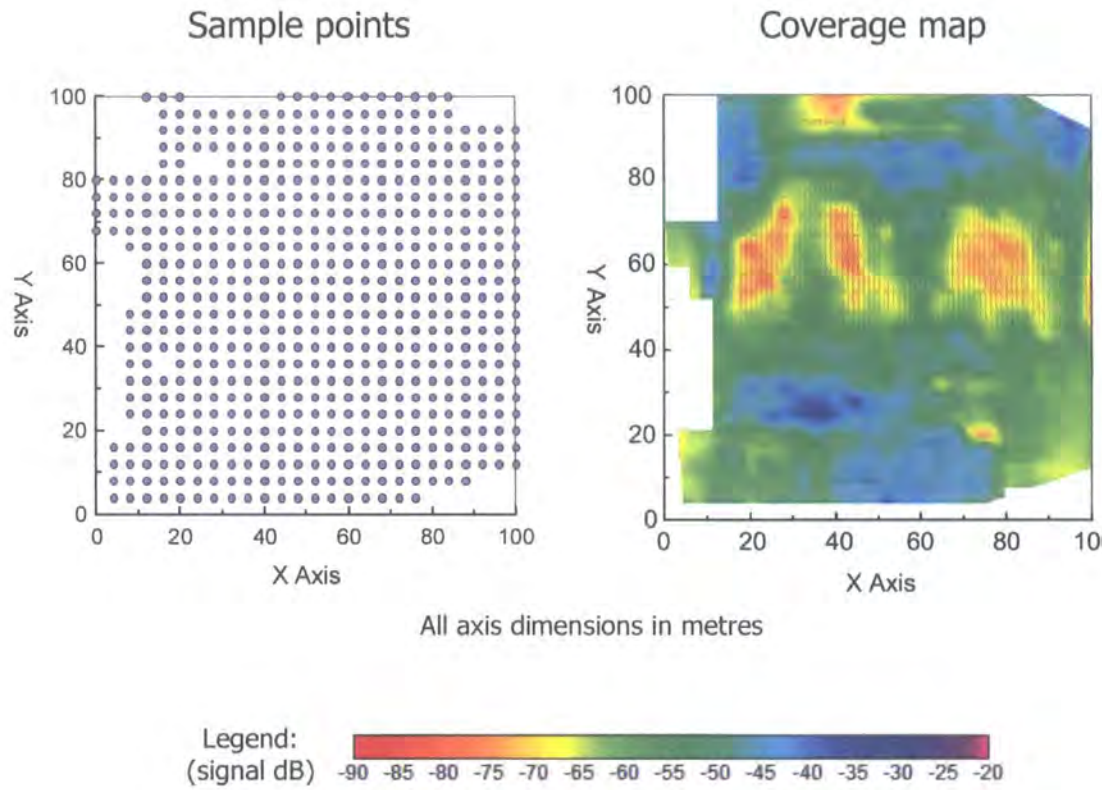

Figure 50: Sample points (left) and associated coverage map (right)

8.1.3 Thin Client Surveying Solution

In performing the survey, the PC performs the more complex tasks:

- Getting the signal strength statistics from the access points
- Storing the data in a database
- Processing the data into coverage graphs

The reason for using a full Windows PC to do the above tasks is that the programming libraries are well suited.

However, the access point can only check signal strengths where the client radio card is located. This means that the surveyor needs to walk around the site with a radio card, which necessitates a computer into which to plug the radio card.

Instead of the surveyor carrying a PC laptop around the site, they instead just carry an RFDT.

The surveyor must control the PC remotely (because the PC gets the signal statistics from the access point), and this requires a remote connection from RFDT to PC. VT Telnet was chosen for this.

The following diagram illustrates how data flows between the ProSurvey PC, access point, and RFDT.

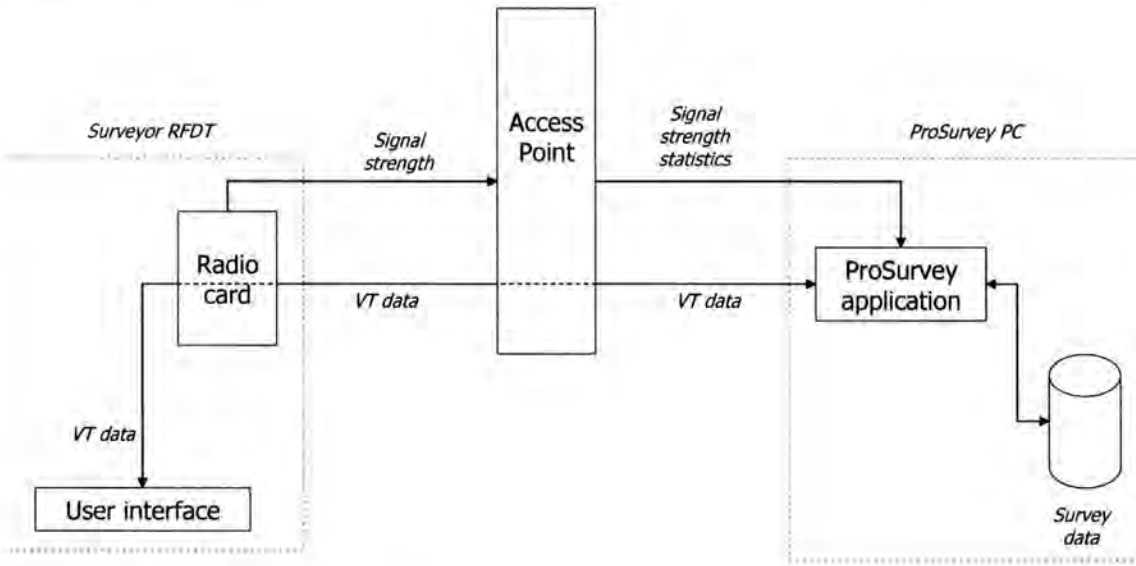

Figure 51: Site survey system hardware interactions

Ignoring the exact hardware interaction, e.g. accepting that the radio card transmits all RFDT information, this diagram can be simplified further to illustrate the system:

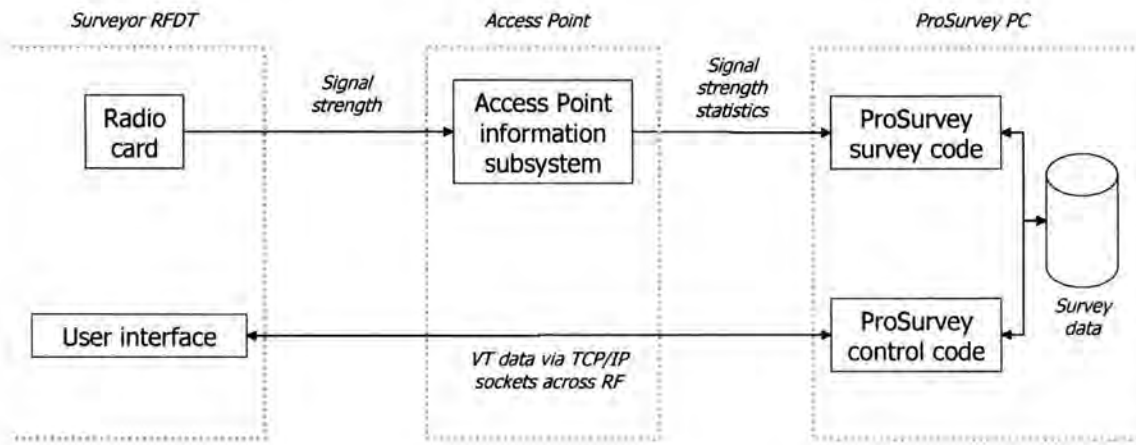

Figure 52: Site survey system data flow

Hence, the survey software uses a thin client approach. The toolkit has much provision for thin client applications using VT, and so was suited for use in the survey software.

8.2 Requirements

The survey software's primary requirements are:

- Handle the VT interface to the survey RFDT
- Obtain signal strength statistics from the access point
- Store and process the results

8.3 Prototype Applications

The survey software was initially created for internal use at Castle Auto I.D. Solutions, and was only a small application to test the viability of a scientific site survey. After proving to be a useful tool, it was subsequently re-written several times. The final design was from the ground-up with the intention of being sold as a commercial application.

8.3.1 Pre-Toolkit Prototype

This was a small application to prove the concept of a grid-based site survey, with an RFDT as a data collection device and a PC to store the tests. The PC software consisted of a DOS application which simply performed network response tests ("pings"):

```
Network Test VT220 Telnet Server
(C) Castle Auto I.D. Solutions, 2001

Socket interface:  Started
Create daemon:     Successful
Listening:         OK
```

Transaction	Location	Average	StanDev	Tx	Rx	Diff
OnAccept 1						
DoPing....	01,01	000	000.0	128	128	000
DoPing....	02,01	000	000.0	128	128	000
DoPing....	03,01	000	000.0	128	128	000

Figure 53: RF Network Test Survey Software V1, PC screen

The VT screens handled by this early survey software were as follows.

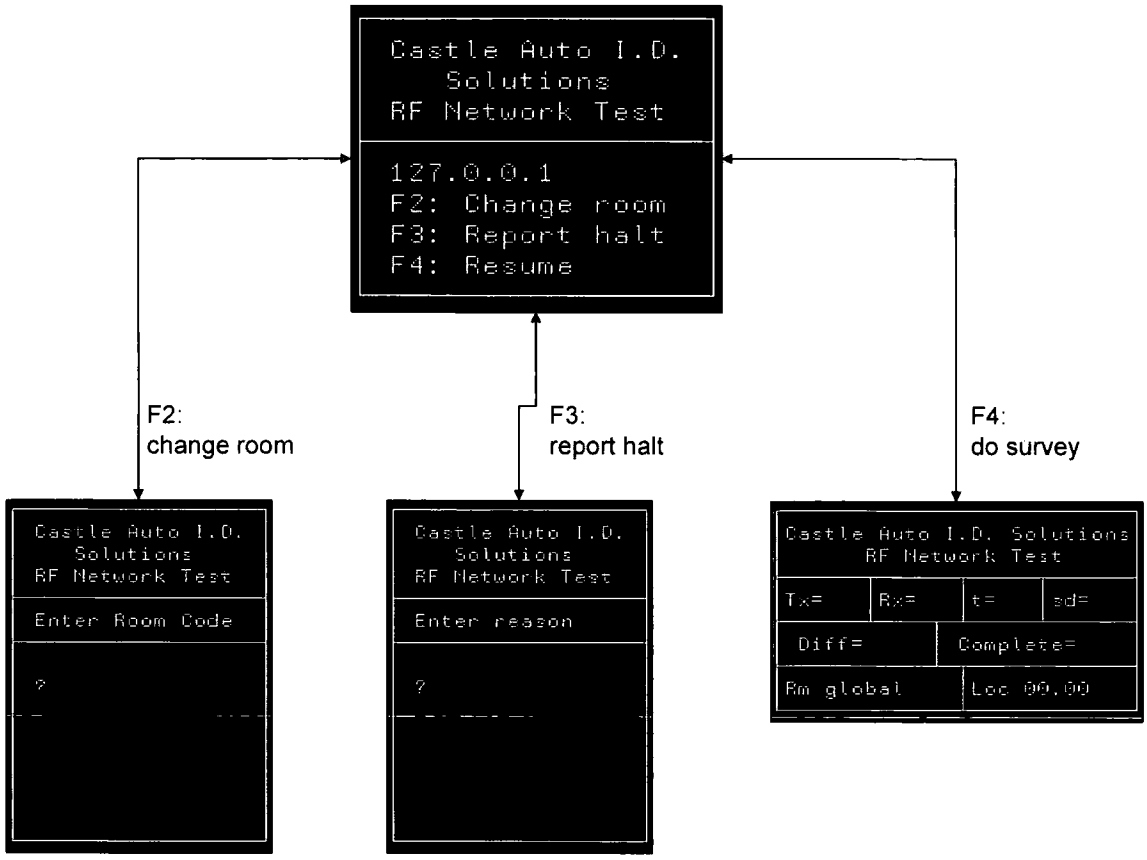

Figure 54: RF Network Test Survey Software V1, VT screens

Although these look similar to the Telnet COM Server output, this application pre-dates the VT part of the toolkit. Hence, this early version of the survey software was a development test-bed for the toolkit, rather than being developed using the toolkit.

8.3.2 Toolkit Developed Prototype

The concept of the survey software had proven commercial success, with the first prototype. Following this, a ground-up rewrite was approved by Castle Auto I.D. Solutions.

In the time which had elapsed since the first prototype, the RF toolkit had been used to develop the VT COM Server. Therefore, the new toolkit-developed survey software could make use of that VT COM Server code.

Each of the input, scrolling list or display fields is controlled by an instance of the VT COM Server code, in the following screens:

Figure 55: RF Network Test Survey Software V3, VT screens

The PC part of the software was implemented using Windows, drawing from sections of the toolkit which will be detailed later.

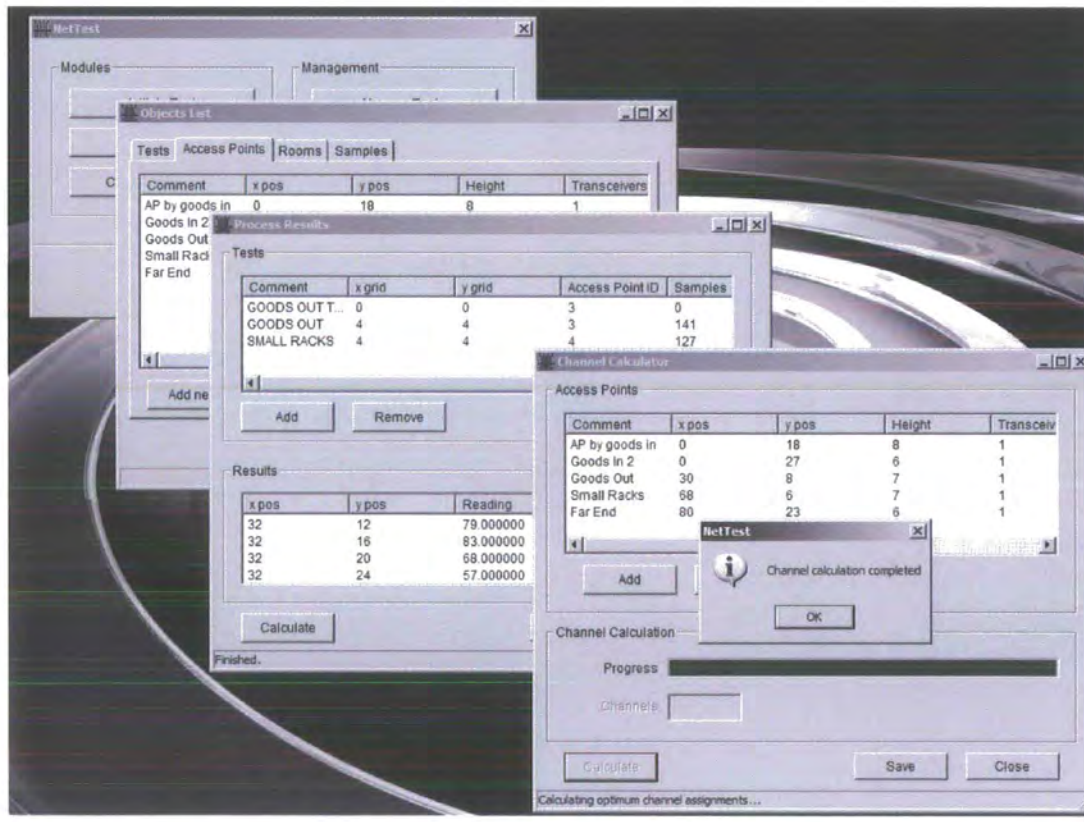

Figure 56: RF Network Test Survey Software V3, PC screens

8.4 Commercial Application

Once it was apparent that there was a market for this new approach to wireless surveying, Castle Auto I.D. Solutions decided to re-design the software as a commercially available off-the-shelf surveying solution, marketed under the name of "ProSurvey."

This final instance of the survey software draws the following sections of the toolkit:

- VT COM Server code, for formatting the VT screens on the RFDT (see section 5.2.2)
- ADO database code, for storing data (see section 5.2.4, claListObject)
- Window position saving and resizing code (see section 5.2.3, claWindowSize)

8.5 Design

For a ground-up rewrite of the survey software, the first considerations were:

- What data should ProSurvey collect and store?
- What were the interrelationships between those data, for storage in a database.

These considerations will now be detailed in order.

8.5.1 Collected Data

The following basic types of data were identified, for any given site survey:

- Samples: a measurement of signal quality, taken at a given point on the site
- Access Points: a record of the AP location which was tested
- Tests: a survey performed on a given AP, under a given set of conditions, recording many different Samples.
- Rooms: to help break the site down into manageable sections, each of the above data types (Samples, Access Points, Tests) is located within a Room.

8.5.2 Database Structure

The requirements for collected data were analysed, and a data structure for ProSurvey was developed accordingly. The structure, shown in the diagram below, shows the various database tables which were designed to hold the survey data.

Each of these tables will be represented by a class, inherited from the toolkit class claListObject, as described in section 5.2.4.

In the database structure diagram, relationships between fields and tables are indicated, with the one-to-many relationships shown as arrows.

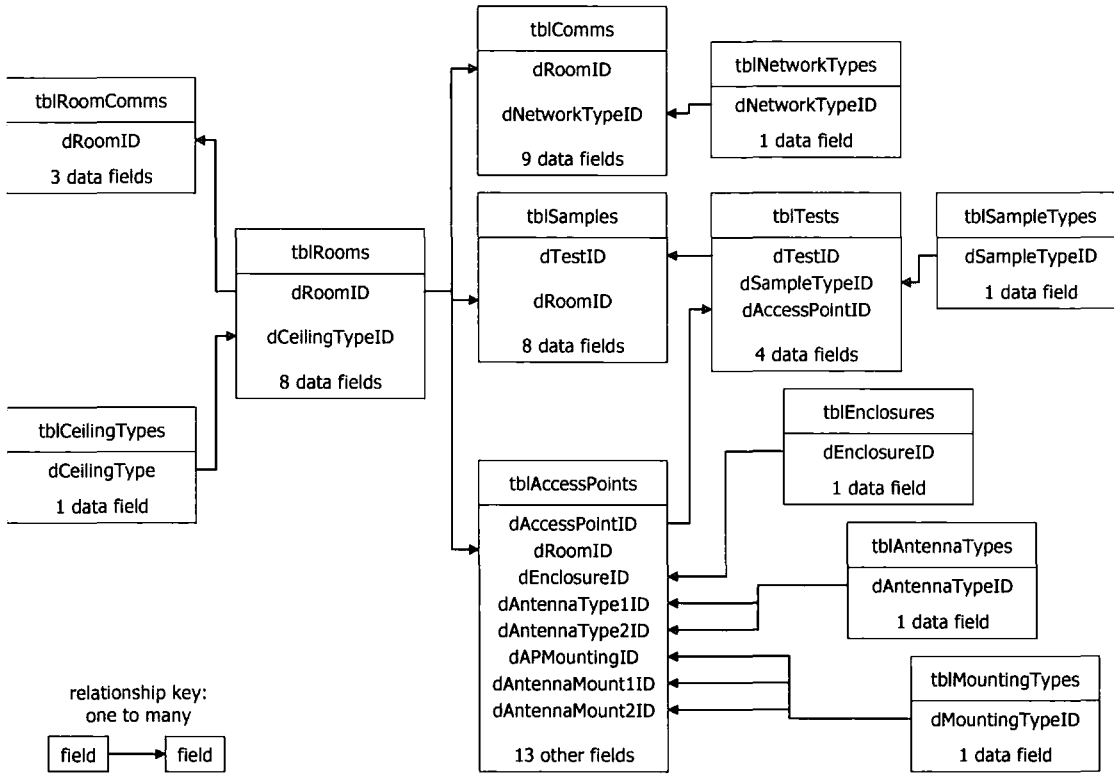

Figure 57: ProSurvey database structure, showing fields with relationships to other tables

8.5.3 Screen Design – VT Thin Client

To collect the radio quality samples, the user must carry an RFDT running a VT Telnet session, connecting back to the ProSurvey PC.

These VT screens will be produced by the VT section of the toolkit, as per section 5.2.2.

The VT screens were needed to perform two tasks:

- Record sample data, by allowing the user to record their position within the room, and show the sample data;
- Change to a different room.

8.5.3.1 Record sample data

This screen must input the user's location (x and y) within the room, and display the radio statistics for the previous sample (signal, noise, signal/noise ratio, bandwidth)

8.5.3.2 Change room

This screen must provide a list of available rooms in a scrolling listbox. The scrolling listbox code was already contained within the toolkit.

8.5.4 Screen Design – Windows PC

The complex tasks of entering site information, recording results to a database, analysing results, and drawing graphs, is performed on the PC.

In order to allow the user to input and process data for a survey, it was decided that the following sections would be required. They are given in the order by which a user would typically use them.

Class name	Purpose
dlgNetTest	Main dialogue
dlgObjects	Display all the rooms, APs, tests and comms which are present in the survey, and enable the user to add, delete or edit those objects
dlgNewRoom	Input a new room on the site
dlgNewAP	Input a new AP location, which the user will survey with
dlgNewTest	Set up a test to be performed, with a given AP under a set of given survey conditions
dlgNewComms	Input a comms location
dlgTest	Allow a test survey to be performed
dlgProcess	Process one or more tests, to analyse surveyed coverage
dlgGraph	Draw graphs of survey data
dlgChannels	Calculate optimum channel allocations for a set of APs
dlgReport	Output graphs for the survey report document, and run the report generation wizard

Table 21: Dialogue/Screen classes

8.5.5 C++ Class Design – Database Access

ProSurvey was structured around the database, as described in section 8.5.2. Bespoke classes for every database table were created, inheriting from the toolkit ADO Database class "claListObject".

This data-centric approach to the design of ProSurvey is important. The inheritance from the toolkit class claListObject is given below.

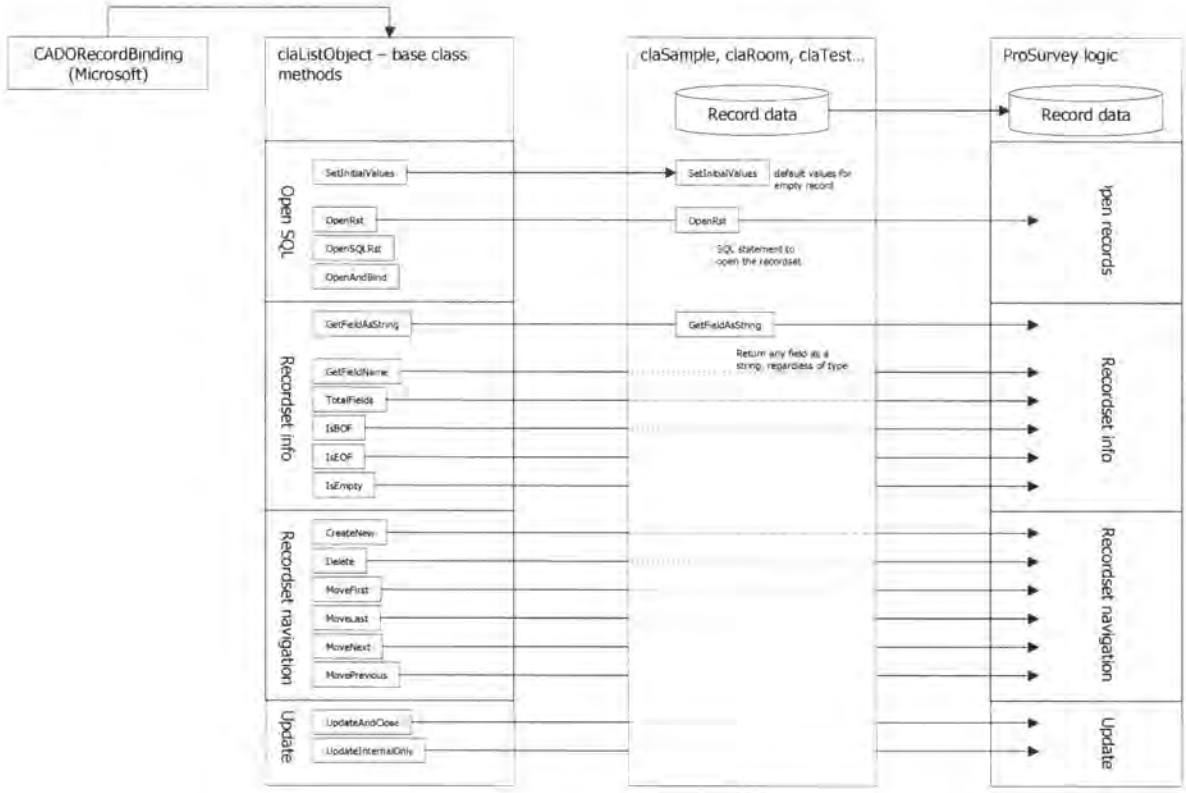

Figure 58: claListObject and derived data encapsulating classes

Because the majority of the work is done by the toolkit claListObject class, encapsulating each database table within its own C++ class is a very straightforward task.

8.5.6 C++ Class Design – Function Calls Between Classes

The links between the classes are shown below. Classes are grouped together according to which section of ProSurvey they are in, i.e. with which dialogue box (window) they are concerned.

The diagrams illustrate which C++ classes call methods in which other classes. The direction of the arrows show function calls, i.e. the flow of application execution.

8.5.6.1 Main Dialogue with links to other dialogues

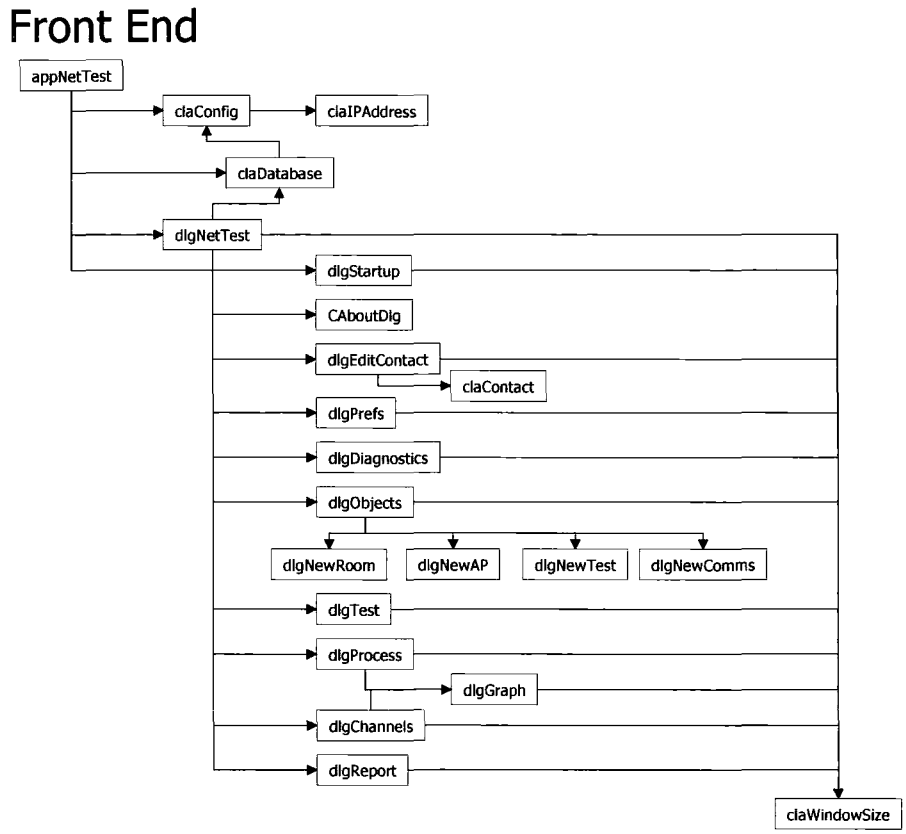

Figure 59: Flow of function calls between classes, main dialogues

8.5.6.2 Enter Room, AP, Test, and Comms data

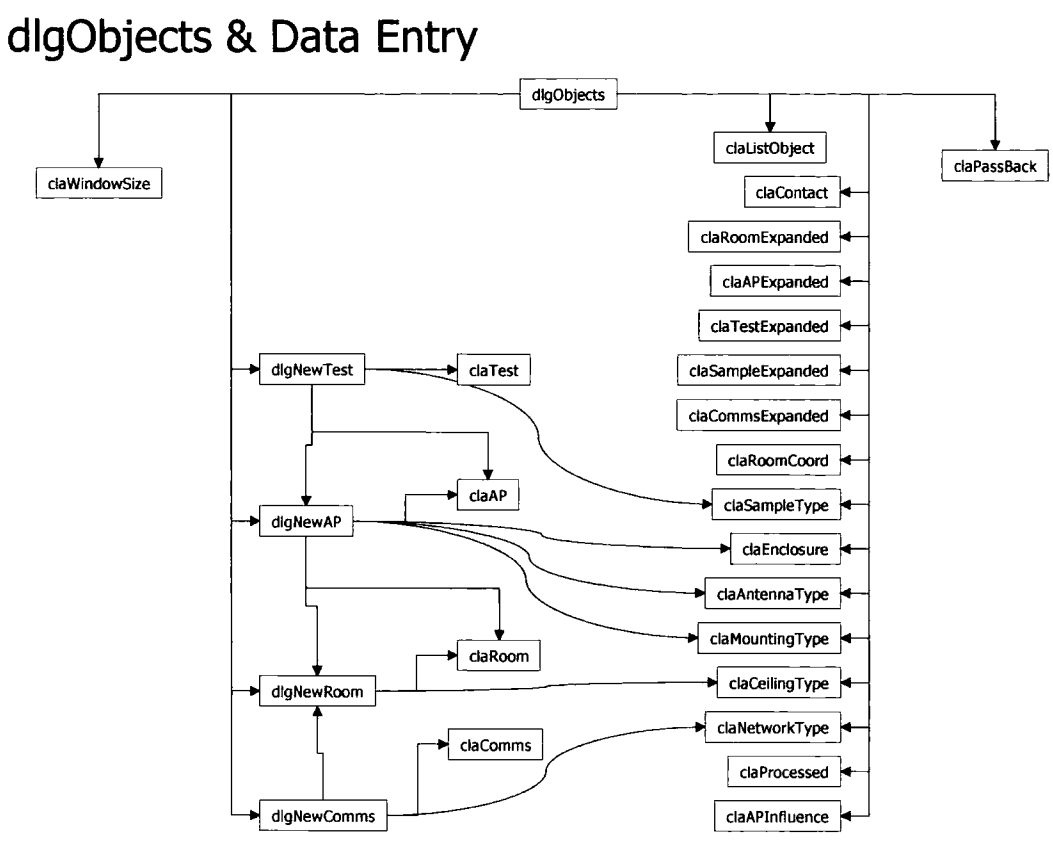

Figure 60: Flow of function calls between classes, object data entry

8.5.6.3 Perform Test

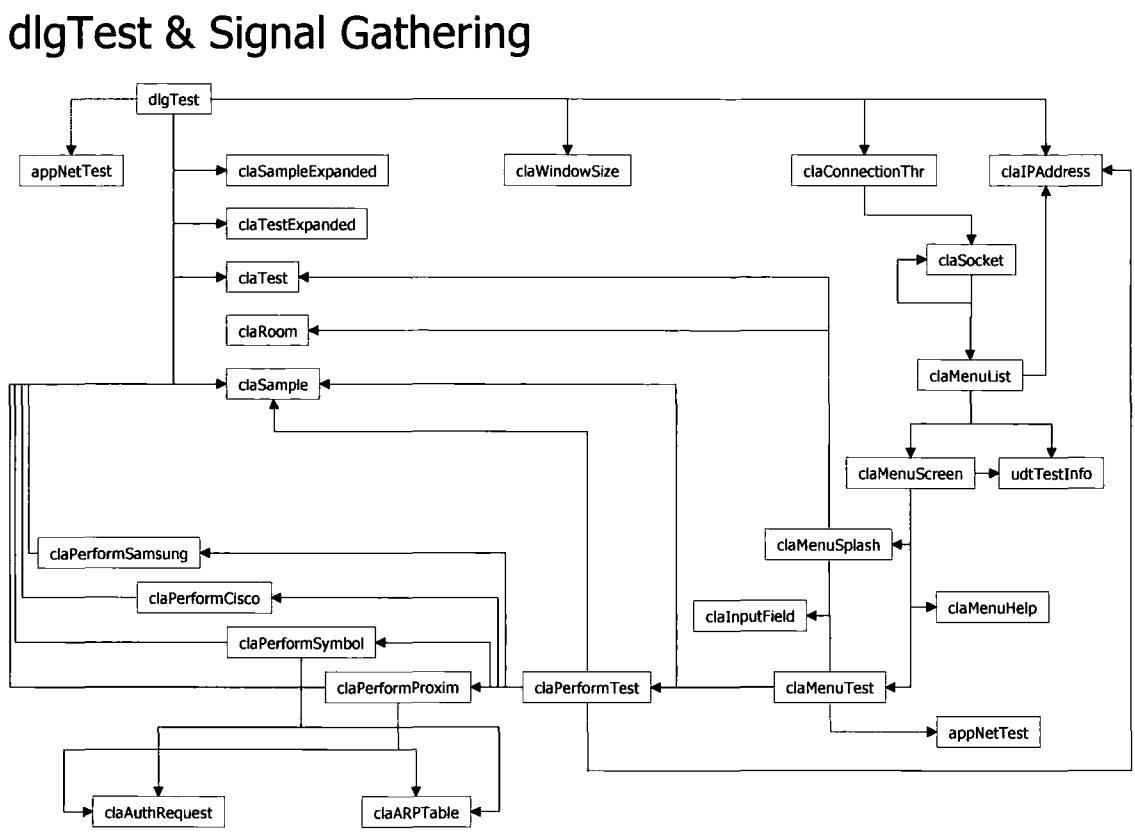

Figure 61: Flow of function calls between classes, survey tests

8.5.6.4 Process Tests

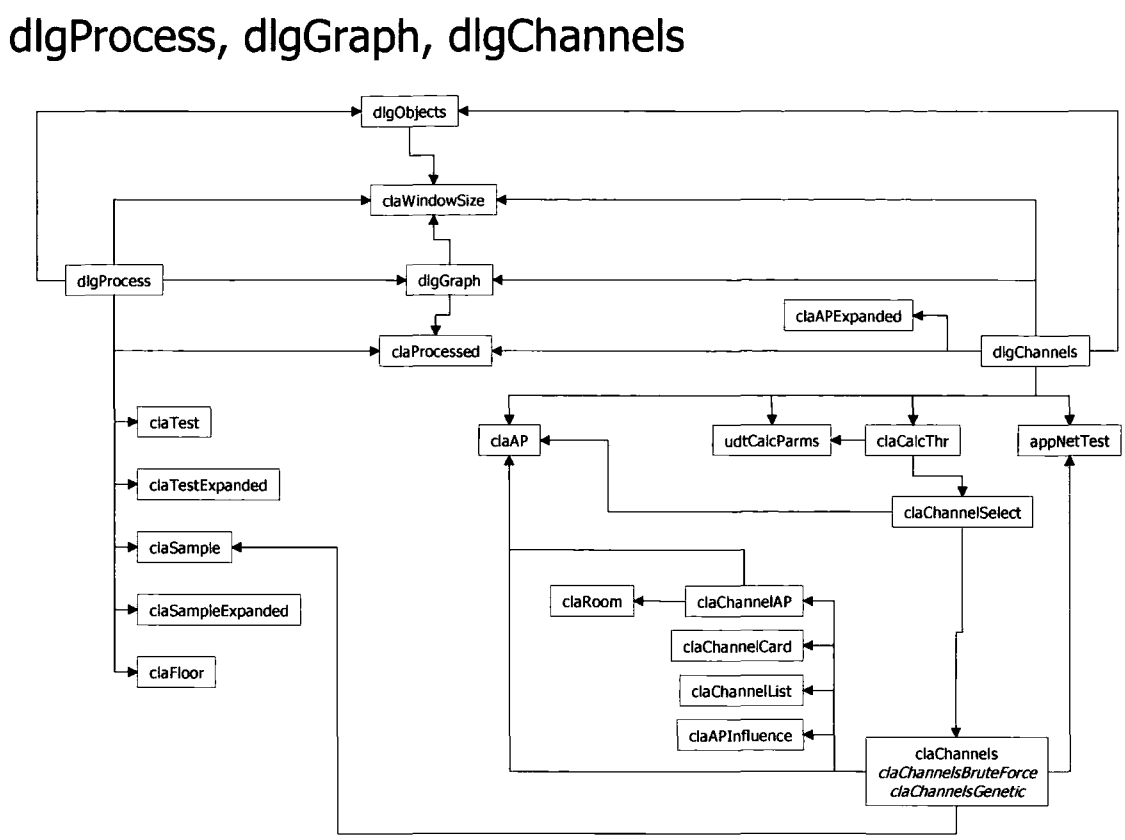

Figure 62: Flow of function calls between classes, processing and channel calc

8.5.6.5 Report Output

dlgReport

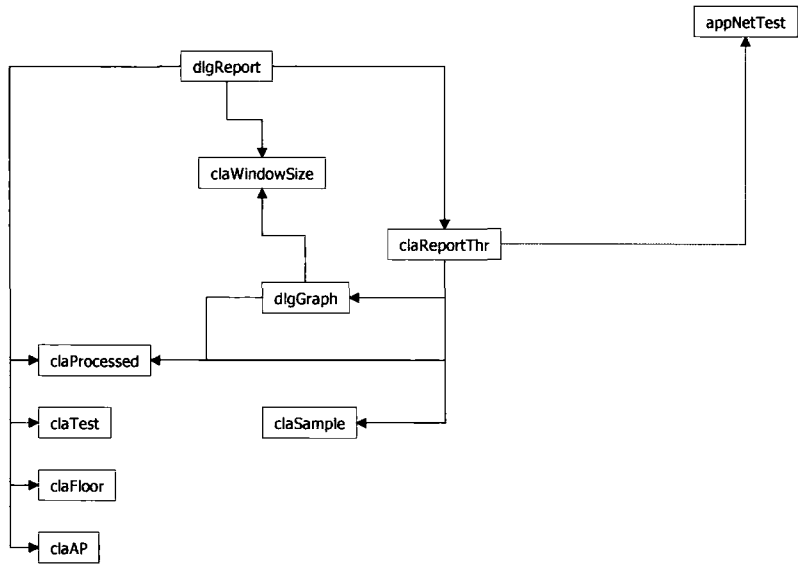

Figure 63: Flow of function calls between classes, report output

8.6 Implementation

8.6.1 Screen Layout – Windows PC

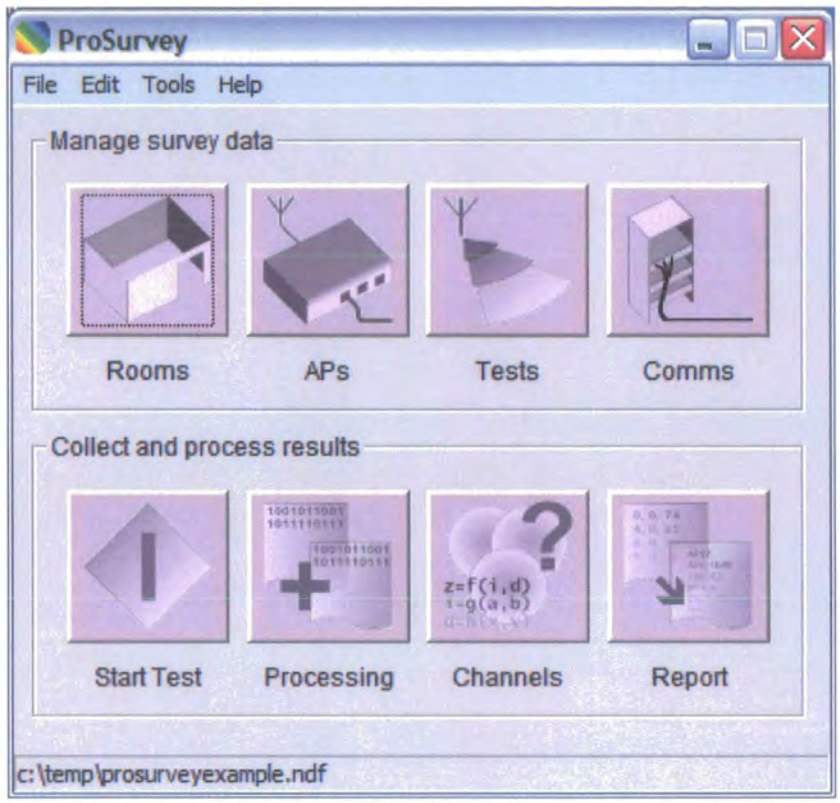

Figure 64: dlgNetTest

Purpose: allow the user to select the different sections of the ProSurvey software.

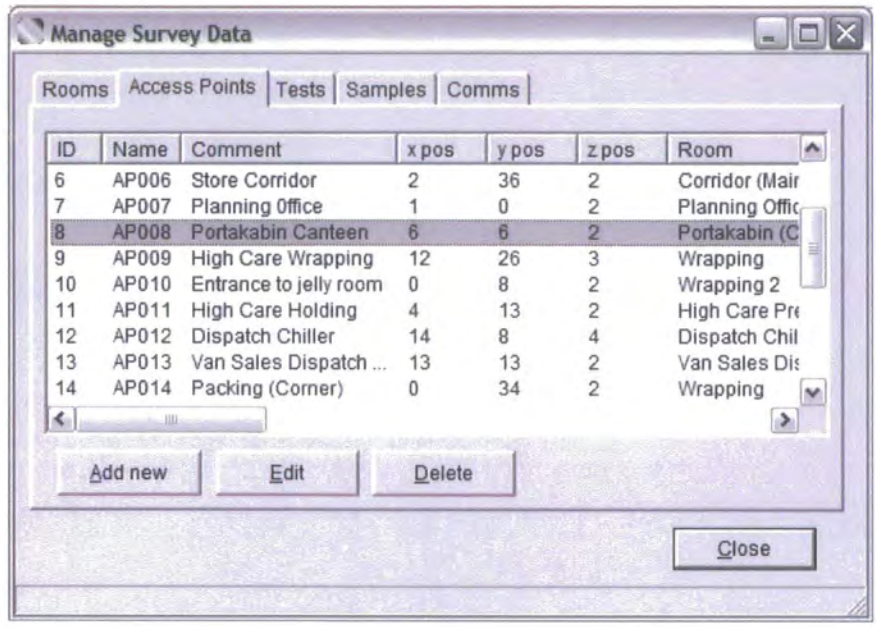

Figure 65: dlgObjects

Purpose: allow addition, editing and deleting of the survey data, from the categories of Rooms, Access Points, Tests, Samples (delete only), and Comms/Network cabinets.

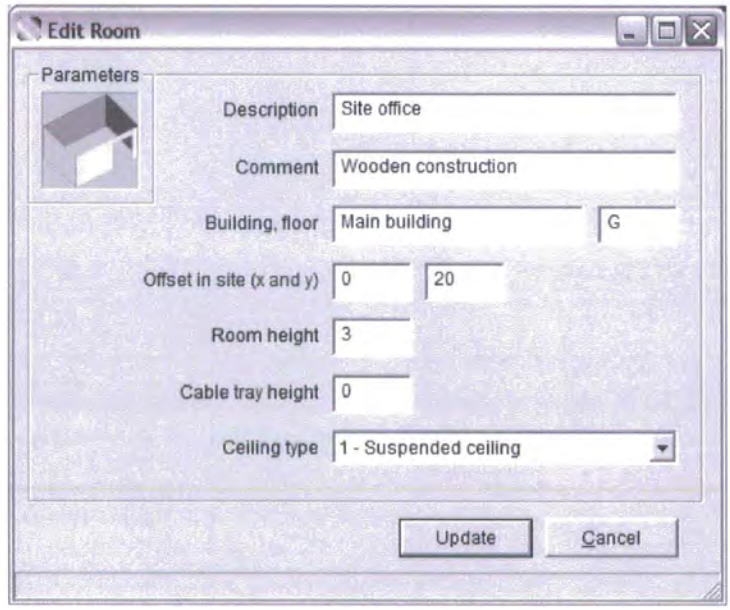

Figure 66: dlgNewRoom

Purpose: define a new room within the site. This screen is selected from the "Manage Survey Data" screen as above.

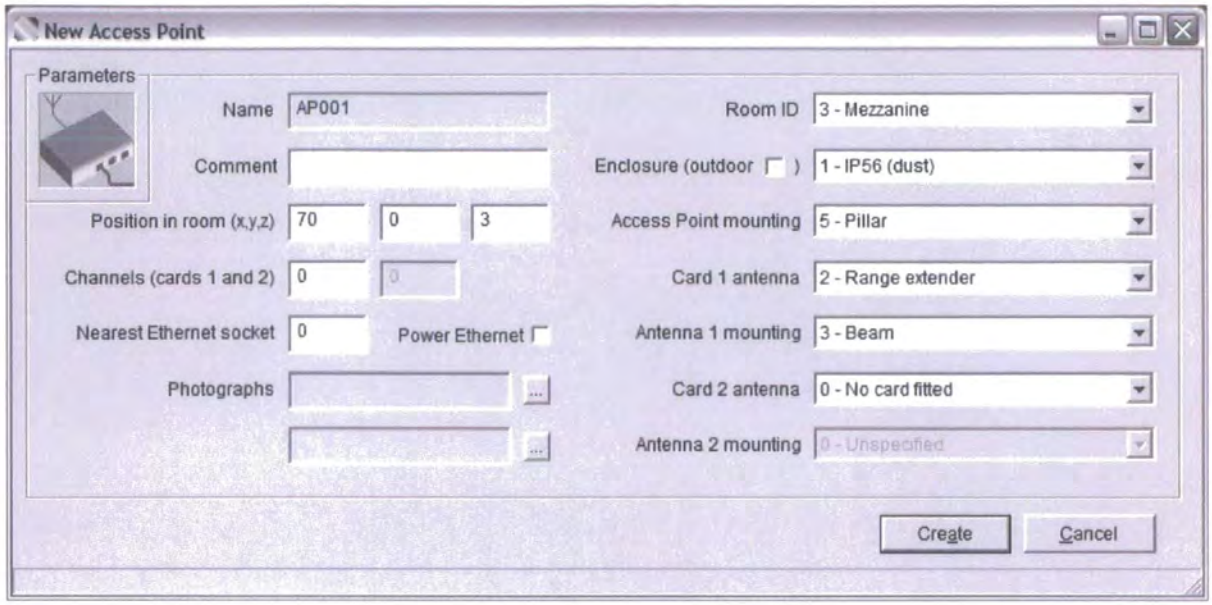

Figure 67: dlgNewAP

Purpose: define an Access Point to be used for the surveying. Depending on the outcome of the survey test, this AP may or may not be included in the final wireless installation.

The AP is located in a given room, which must have been defined first.

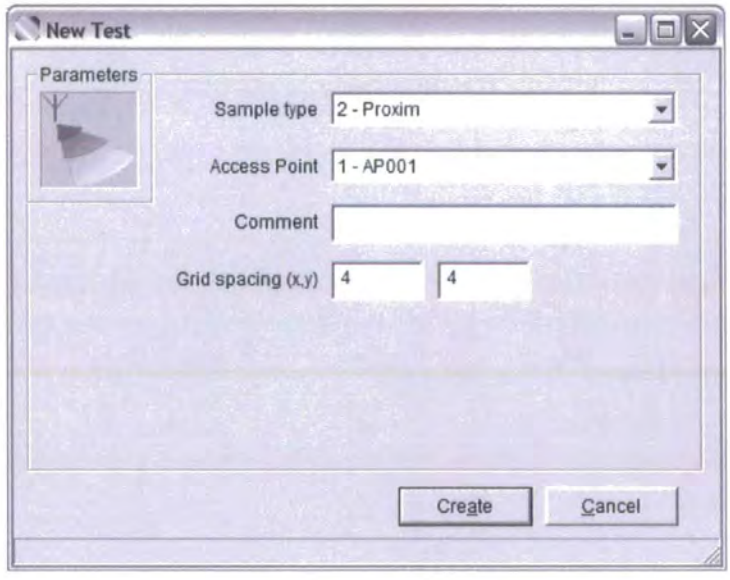

Figure 68: dlgNewTest

Purpose: add a new test. A test ties a group of sample data to one particular Access Point, and one survey with that AP.

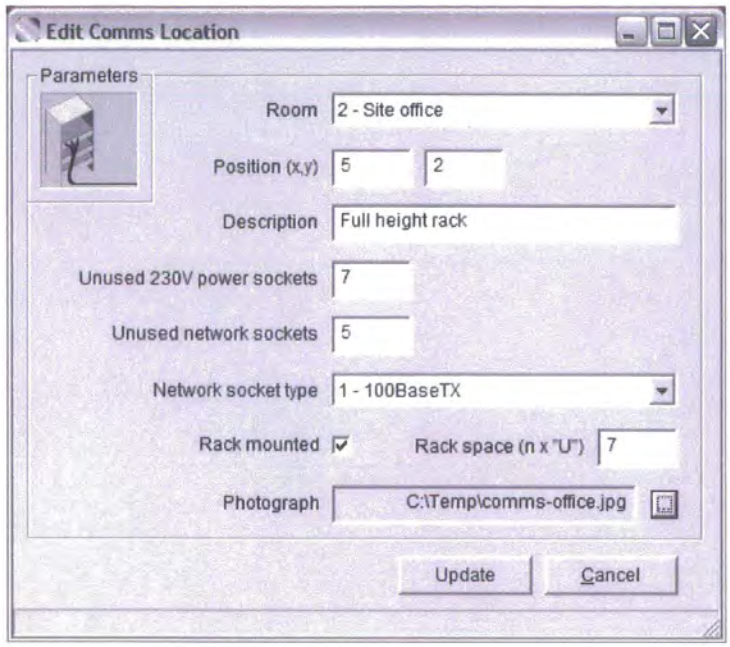

Figure 69: dlgNewComms

Purpose: edit a comms cabinet / network cabinet location. The comms location is within a given room. Provision is made for including a photograph.

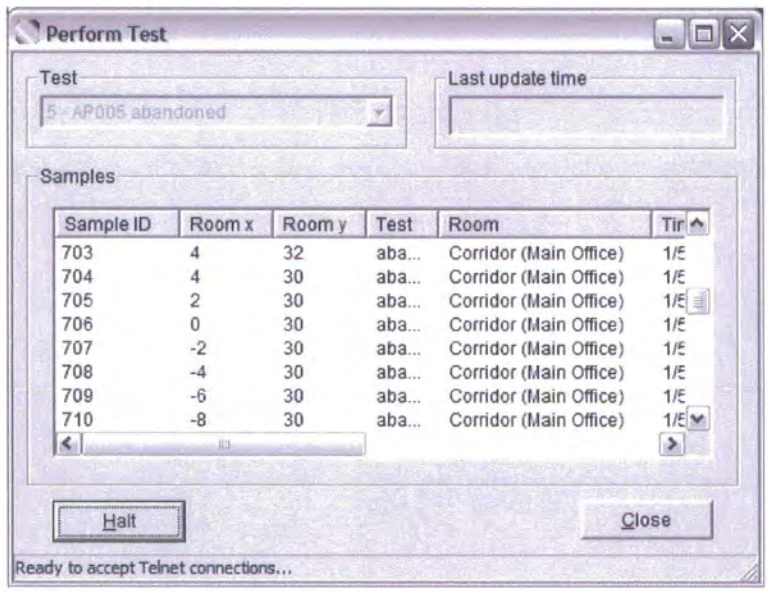

Figure 70: dlgTest

Purpose: once a test has been defined, with a given Access Point, the user can start surveying. This window shows a survey test being run, showing the existing sample data for that test.

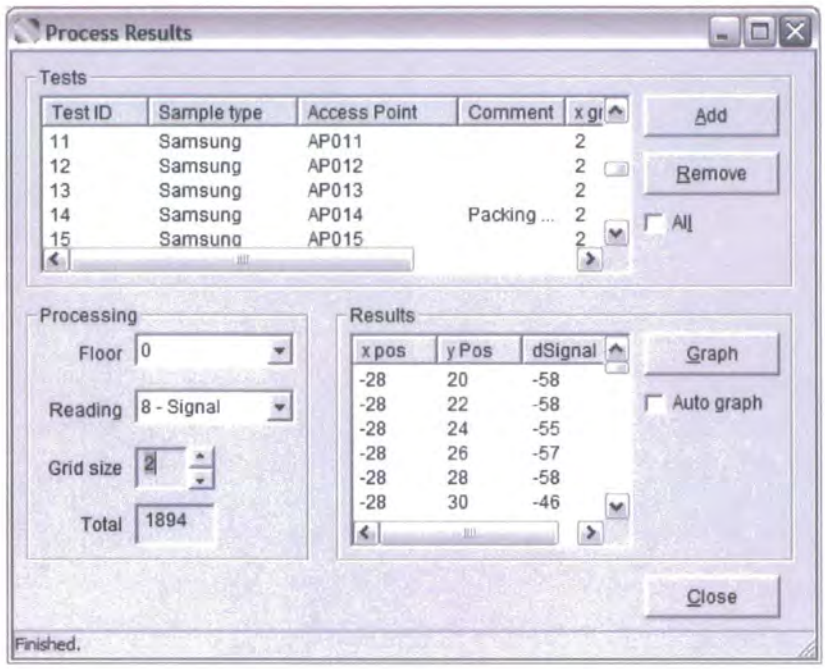

Figure 71: dlgProcess

Purpose: allow the user to process the survey test data. Any number of tests can be processed together, using the Add/Remove buttons. By selecting a floor (because the rooms have floors), and a reading type (Signal/Noise Ratio, Signal, Noise, Bandwidth), the Results section is updated to show the combined results.

By clicking on Graph, the combined results for those tests can be shown graphically.

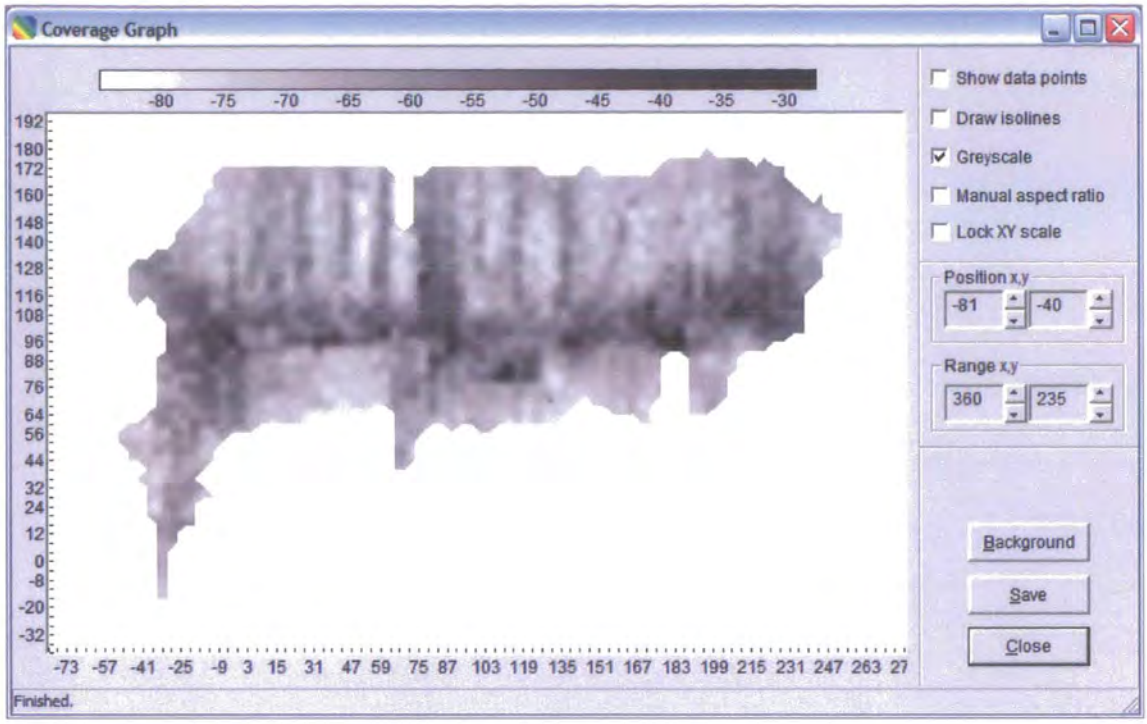

Figure 72: dlgGraph - greyscale

Purpose: show graphically the signal quality for survey tests, either singularly or in combination. This particular diagram shows how the signal (black) propagates vertically along rows of racking, which come off a central corridor through the site. The results can also be shown as a full-colour spectrum instead of greyscale:

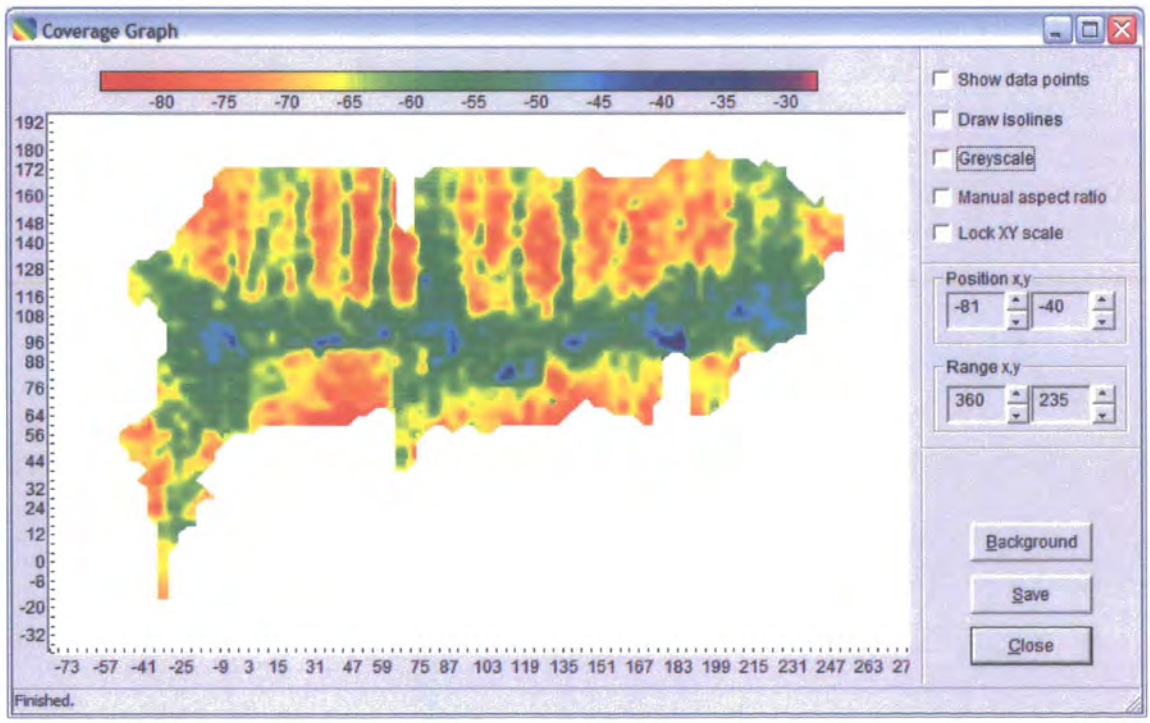

Figure 73: dlgGraph - colour

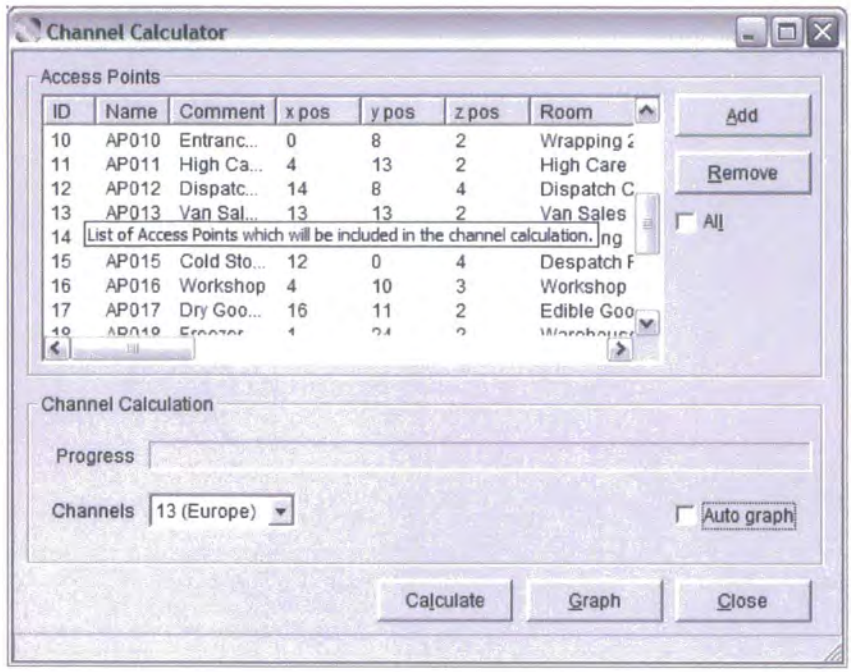

Figure 74: dlgChannels

Purpose: allow the optimum allocation of channels to be calculated. The user selects how many channels they wish to make use of, and then a separate thread runs to perform the calculations. This is quite a lengthy process; hence, there is a progress bar.

8.6.2 Screen Layout – VT Client

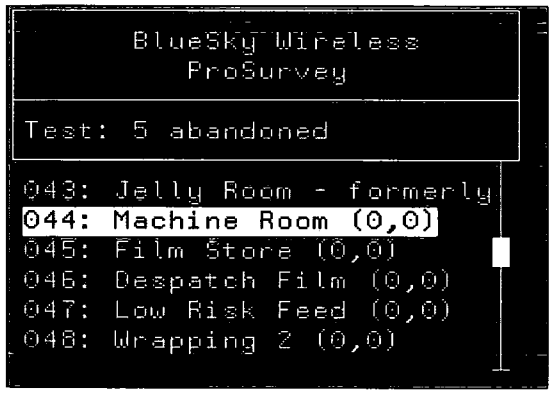

Figure 75: claMenuRoom

Purpose: allow the user to select which room they are entering. This means that they need only know their location within their room; ProSurvey can then calculate the offset, based on data provided when the rooms were defined.

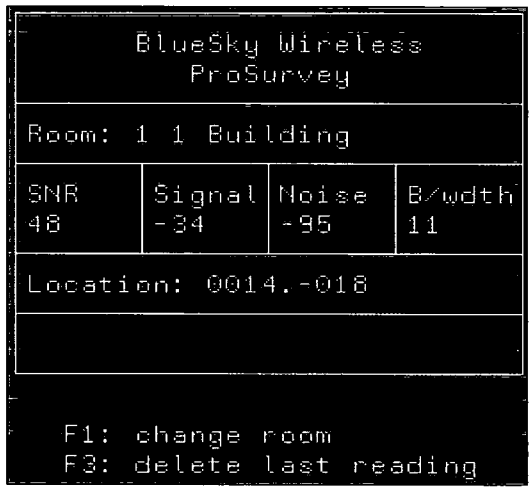

Figure 76: claMenuTest

Purpose: allow the user to type in their location within the room, and press Enter to cause ProSurvey to take a reading/sample at that grid location. The readings for that location are then returned to this screen, for the user to make a visual check of the signal.

8.7 Real-world Surveys

In order to demonstrate the usefulness of ProSurvey, and therefore evaluate the benefits which have arisen from this thesis' toolkit, some actual site surveys will be examined.

These are data which have been collected by Castle Auto I.D. Solutions, during their time using ProSurvey to perform commercial RF site surveys.

8.7.1 Racking

As detailed at the start of this thesis, many commercial applications centre on warehousing. This often includes "racking", i.e. tall, continuous rows of shelving which may be many stories tall.

The survey graph below is real data from a survey of outdoor racking:

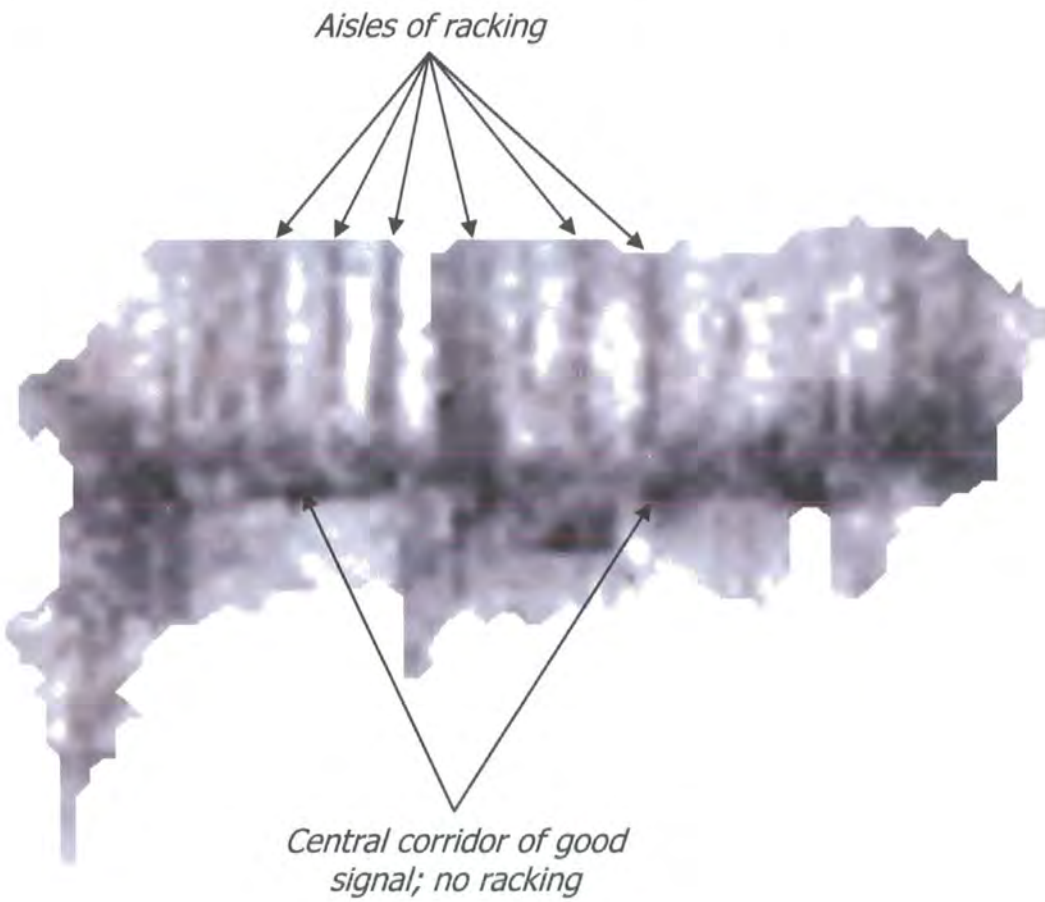

Figure 77: Survey graph illustrating the effects of racking

The above graph shows clearly a central line of good signal (black) running left to right. This is a "corridor" through the site, where there is no racking. Signal is not inhibited.

Coming off the central corridor are lines of racking, extending towards the top of the site. The RF signal is seen to be excellent between the racking, where it is being reflected along the aisles.

This site is particularly illustrative for the phenomenon of racking because it is outdoors. There are no extraneous signals being reflected down off a roof, which can blur these effects on a graph.

8.7.2 Noise

ProSurvey records not only the signal strength, but also the noise.

In all of the sites which Castle Auto I.D. Solutions have surveyed, noise has been almost constant throughout the site.

However, there was one exception, and that is reproduced below.

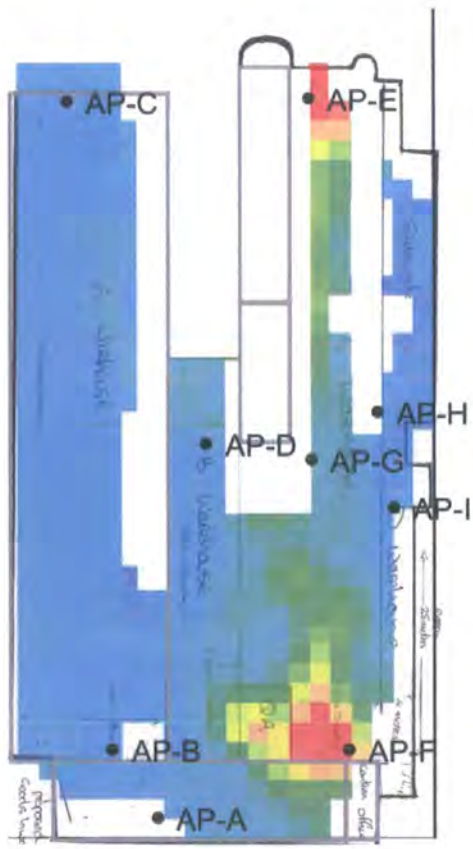

Figure 78: Survey graph illustrating noise

Note that this graph has the customer's site map overlayed onto it. In this survey, the noise adjacent to Access Point "F" was so strong that the Signal/Noise ratio fell below acceptable levels even close to the AP, despite the actual signal level being high.

This illustrates a key selling point of ProSurvey, for Castle Auto I.D. Solutions: customers cannot just locate Access Points based upon a visual assessment, as the noise levels can never be predicted.

8.7.3 Signal/Noise Ratio and Bandwidth

ProSurvey records the "bandwidth" of the client, in addition to the physical Signal/Noise Ratio (SNR). The bandwidth is the data rate of the Physical layer (PHY) of 802.11b RF. It has an approximate correlation to SNR, as shown below, although there are factors other than SNR which affect bandwidth.

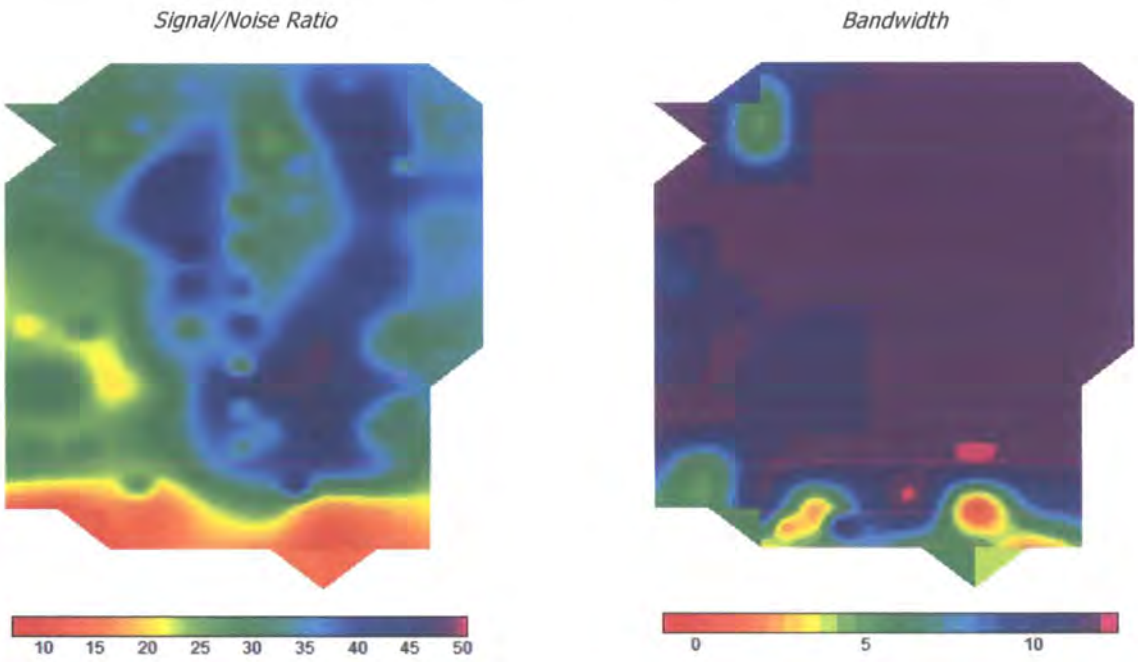

Figure 79: Survey graphs comparing SNR and bandwidth

8.7.4 Speed of Surveying

Despite ProSurvey being a very detailed system, and recording much data, it is important to note that the speed of surveying is still high. The surveyor does not have to spend a prohibitively long time at the site, which would make it financially unviable.

Below is a screenshot of the actual sample data which has been recorded. Each sample is stored with a date and time stamp, and so the time between samples can be seen:

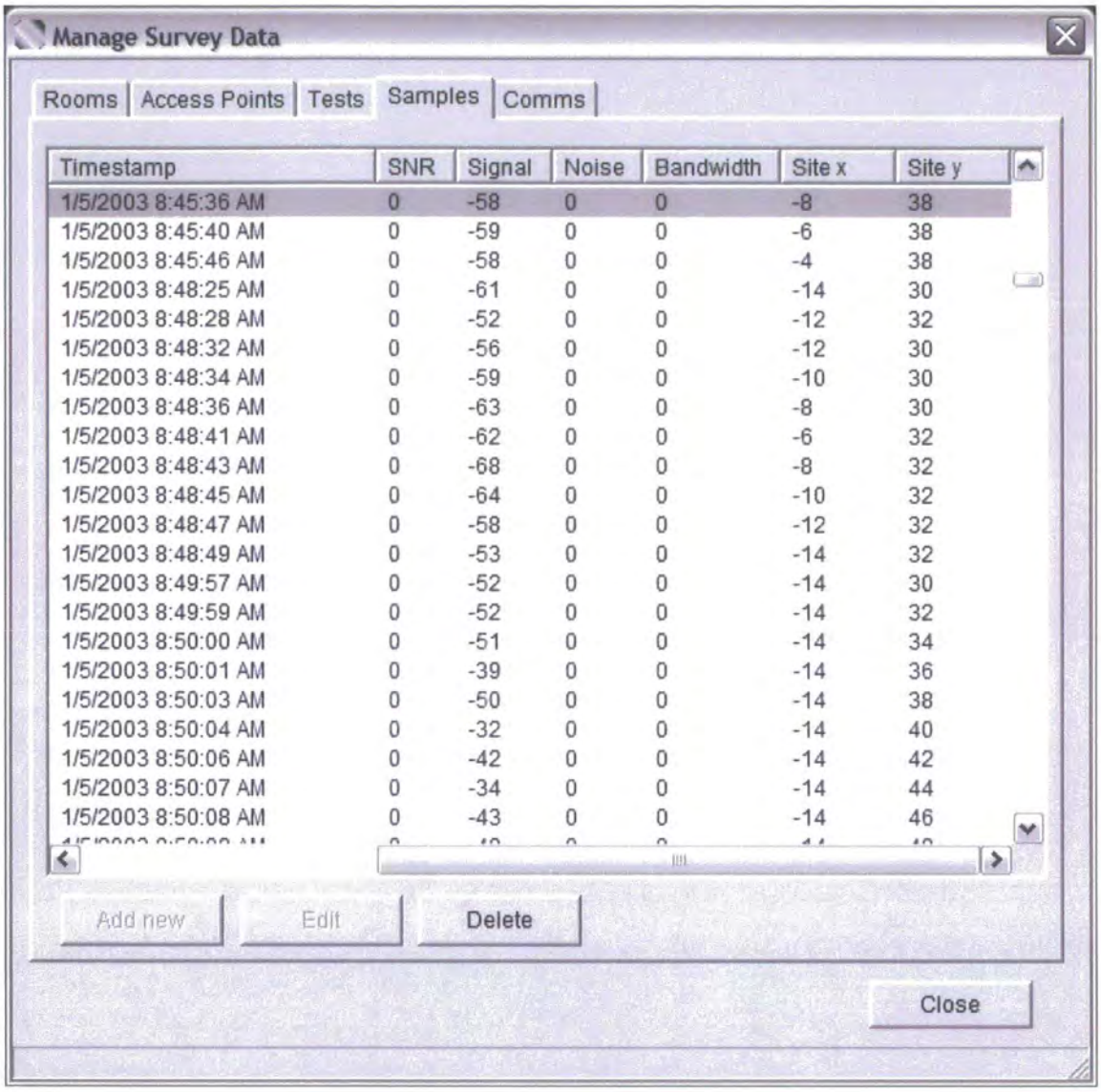

Figure 80: Sample data showing survey speed

The time between samples is just a couple of seconds, for the most part. Once setup time has been considered, this typically results in a 100x200m site being surveyed in less than 1 day.

8.7.5 "De-skilling" Surveying

ProSurvey forces a structured approach to surveying, using the grid system. This means that the surveyor needs less technical knowledge of RF and surveying, to perform a survey to the same detailed standard.

This has been shown within Castle Auto I.D. Solutions, who are now able to use technicians without detailed RF knowledge.

9 Conclusion

This thesis has shown the advantages of a toolkit approach. These advantages will now be summarised.

9.1 The Toolkit for Code Re-use

Several different commercial applications have been developed from the toolkit components of this thesis, and code re-use has played a large part in that. These commercial applications have been sold at a profit by Castle Auto I.D. Solutions, and the fact that code was shared between these projects is a further benefit in reducing development time.

Lower development time means either a lower cost to customer, and therefore more chance of winning business; or it means more profit to the company, by the developer taking fewer paid hours to perform the same task.

Section 6 provides the evidence for this code re-use.

9.2 RF Site Surveying and ProSurvey

9.2.1 Toolkit Re-use

Through developing the toolkit, the author was able to design and implement an RF site survey system at a reduced cost to Castle Auto I.D. Solutions.

If toolkit code was not available, it is possible that the company could not have justified spending the developer's time on software for such a radical (and initially un-proven) surveying technique.

However, the toolkit code was available, and developing ProSurvey (earlier called NetTest) fed back into developing the toolkit.

Had the toolkit not existed, and ProSurvey not been created, Castle Auto I.D. Solutions would not have a unique selling point in the marketplace of RF surveys and installations.

9.2.2 Commercial Sites Surveyed

Central to winning RF business is the structured nature of the RF survey, as detailed in this thesis. This enabled Castle Auto I.D. Solutions to show customers what their RF coverage would be, and guarantee the success of that RF installation.

This has given Castle a unique position in RF surveying. No site has ever required re-visiting following an installation, for remedial work. This is in contrast to other companies in the RF network business, who frequently have to perform remedial work following a less structured form of surveying.

At the time of writing this thesis, at least 20 sites have been surveyed within the last 12 months using ProSurvey. This represents a five-figure sum on survey consultancy charges alone, and many more times that in Access Point sales and installation charges.

9.2.3 ProSurvey as an Off The Shelf Package

Castle Auto I.D. Solutions has released ProSurvey as a commercial package. At the date of writing this thesis, a major European network distribution company (Anixter) has been signed to sell ProSurvey in the UK, and with possibilities of rolling it out internationally. Hewlett Packard is currently in negotiations as a partner, with a view to approving it as their preferred survey method.

9.3 Future Work

ProSurvey has proven to be the most successful product of the toolkit, and has attracted significant interest from major RF network manufacturers. This includes Hewlett Packard, Proxim, and Symbol.

There are two fields within RF surveying and ProSurvey which could form the basis of future work.

9.3.1 RF Coverage Simulation

Some of the wireless hardware manufacturers have asked for ProSurvey to predict RF coverage, based upon a survey: perhaps moving an Access Point around the site using a software model, and simulating what the effect might be on coverage.

There is significant scope for development of such a model, beyond the boundaries of this thesis.

9.3.2 Channel Calculation

The RF channel calculation algorithm which is used in ProSurvey may be the subject of further work. It is not described in this thesis, because it is not relevant to the work of the toolkit. However, the task of solving this n-factorial permutation problem can be examined further.

9.4 Final Conclusion

The benefit of the toolkit approach to producing bespoke RF software has been demonstrated. This has been achieved within a commercial environment, with a private company partly funding the project. The commercial success of the company in this area quantifies these benefits.

The most significant product created with the toolkit has been the RF survey system. This has revolutionised how sites are surveyed prior to equipment installation. Major hardware manufacturers have shown much interest in the system, and it is now being sold by several vendors throughout Europe.

References

[1] "Castle Auto I.D. Solutions" and "BlueSky Wireless" are trading names of Castle Labelling Ltd, 15A Redwell Court, Harmire Enterprise Park, Barnard Castle, County Durham

[2] http://www.tcsonline.org.uk/

[3] "Reading Between The Lines", Datalogic Corp., 1998

[4] "Reading Between The Lines", Datalogic Corp., 1998

[5] "Computer Networks (3rd edition)", p. 28, Tanenbaum, Prentice-Hall International Inc 1996

[6] "Computer Networks (3rd edition)", p. 36, Tanenbaum, Prentice-Hall International Inc 1996

[7] IEEE 802.3 CSMA/CD (ETHERNET): http://grouper.ieee.org/groups/802/3/

[8] IEEE 802.11 Main General Info Page: http://grouper.ieee.org/groups/802/11/main.html

[9] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.28, IEEE, 1999

[10] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.216, IEEE, 1999

[11] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.18, IEEE, 1999

[12] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.3, IEEE, 1999

[13] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.4, IEEE, 1999

[14] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.42, IEEE, 1999

[15] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.216, IEEE, 1999

[16] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.53, IEEE, 1999

[17] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.42, IEEE, 1999

[18] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.62, IEEE, 1999

[19] "Antennas" p6, Blake, Wiley 1966

[20] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.181, IEEE, 1999

[21] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: ANSI/IEEE 802.11 Standard", p.218, IEEE, 1999

[22] http://www.astronantennas.com/unity_pattern.html

[23] http://www.astronantennas.com/11elem_125dbd.html

[24] "Spread Spectrum In Communication", p.10, Skaug & Hjelmstad, Institute of Electrical Engineers, 1985

[25] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.53, IEEE, 1999

[26] "Multiphase Complementary Codes", p.546-552, R. Sivaswamy, IEEE Trans. on Information Theory vol IT-24 No 5, 1978

[27] "Higher-Speed Physical Layer Extension in the 2.4GHz Band: ANSI/IEEE 802.11b Standard", p.45, IEEE, 1999

[28] "A Brief Tutorial on the PHY and MAC layers of the IEEE 802.11b Standard", p.7, http://www.eirp.org/~henty/webtut.pdf, Benjamin E. Henty, 2001

[29] Microsoft Developer Network (MSDN) Library, "Processor Types and Platforms", Feb 2003 edition

[30] "Intermec Trakker Antares 24XX Terminal User's Manual Addendum", p.A-4 & A-5, Intermec Corp, 2000

[31] http://thenew.hp.com/country/us/eng/prodserv/notebooks_handhelds.html

[32] "Dolphin 7400RF Mobile Computer", p.2, Hand Held Products Inc, 2002

[33] Windows CE overview: http://www.microsoft.com/windows/embedded/default.asp

[34] Windows CE 3.0 Kernel Services: http://www.microsoft.com/Windows/embedded/ce.NET/previous/evaluation/features/default.asp

[35] Microsoft eMbedded Visual Tools Product Information: http://msdn.microsoft.com/vstudio/device/prodinfo.asp

[36] VT thin client resource site: http://vt100.net/

[37] 5250 Telnet Interface RFC: http://www.faqs.org/rfcs/rfc1205.html

[38] The X Protocol: http://www.x.org/about_x.htm

[39] Citrix NFuse Demo Room: http://demoroom.citrix.com/nfuse17/web2/nfc.htm

[40] VT thin client resource site: http://vt100.net/vt_history