Analysis, Design and Implementation of a Helpdesk Management System

Mark Knight
Information Systems (Industry)

Session 2004/2005

Summary

This project has been established to fulfil the need at Diagonal Solutions, a software development house, to create a new helpdesk management system.

Helpdesk management is central to many organisations that utilise Information Technology. They ensure that incidents that Users encounter are dealt with by the relevant people as quickly as possible. Diagonal Solutions operate a helpdesk for a number of different clients. Unable to resource a suitable package from the open software market, it has been decided to carry out the work in house.

The project explores different project approaches and methodologies and outlines a number of suitable technologies that could be used to create suitable software. After a requirements analysis is carried out, the decision to design and implement an integrated software solution using Microsoft's ASP.NET and other Microsoft Server technology is made. The results of the implementation are then assessed both in terms of the usability and of the functionality. The system is then evaluated and conclusions on the success of the project are drawn.

Acknowledgements

First and foremost I would like to thank my parents for their continued support throughout this year.

Many thanks to my housemates John, Laura, Chris and Jon for putting up with me and for providing me with much entertainment, I only hope I repaid the favour.

Thank particularly to Mike Lord and Guy Hodges at Diagonal solutions for allowing me back and for providing me with all the resources in order to see the project succeed.

For the help, input and backing of my Supervisor, Pat Hill – thank you.

Finally, recognition must go to James B. Beam Distilling Co. for producing "The World's Finest Bourbon" which I have had the pleasure of enjoying on occasion throughout this year.

Contents

Chapter 1

Introduction

Diagonal Solutions is a mid-sized company providing technology solutions to a number of corporate and government bodies. The need for a new issue tracking system has been identified by the company's management in order to achieve better customer relationships and improved reporting for management. It is also believed that a new system will result in easier software management and improved management of customer accounts.

1.1 Problem Statement

The system currently in place at Diagonal Solutions was produced to fulfil the need to log calls from customers and provide support technicians with details of their problems. Whilst this works effectively, it lacks the functionality that many other solutions provide. The main additional functionality that Diagonal Solutions require is the ability to manage customer contracts – namely their Service Level Agreement (SLA). The SLA is a document that specifies the agreed times for a response and fix to customers issues. With this functionality, management will be able to track how well technicians are performing and ensure that the company's contractual obligations are met. The current system runs as a Windows application that must be installed onto each machine that it is to be used on. As a result, it is a time consuming process to have to install updates and load the software when new workstations are purchased.

1.2 Project Scope

This is an Information Systems project that will cross many disciplines including management, software design, software development and systems architecture.

1.3 Minimum Requirements

By the end of the project the following minimum requirements will have been fulfilled:

- Section on the analysis and evaluation of current systems available to the market
- A full analysis of the requirements at Diagonal Solutions

- Design of the system (including UML design techniques)
- Implementation of a suitable prototype call logging system with the ability to track cases and create reminders for support staff when deadlines approach in line with the clients individual SLA.
- Evaluation of the prototype with respect to the users' requirements

1.4 Further Enhancements

Further enhancements could be made to the system prototype and will be implemented depending on the time available.

- Further development based on the evaluation of the prototype;
- Integration with Diagonal Solutions' time recording system;
- Ability to store details of and assign calls to a third party engineer;
- Management reporting – enabling managers to obtain information from support calls such as response time and time taken to fix issues;

Chapter 2

Project Approach

This section outlines the need for a structured approach in order to achieve a successful project.

2.1 Project Management

Before starting any project it is important to be clear of the outcomes that are expected and the processes involved that will achieve those outcomes. There are a number of approaches that can be taken in order to manage the resources that are required to complete a successful project. In [1], Hughes and Cotterell outline the activities to be carried out in project management as:

“ 1. The feasibility study
2. Planning
3. Project execution”

When faced with this particular project one finds oneself in a unique position. Whilst it is an academic assignment, it is also a live project that brings genuine business benefits to Diagonal Solutions. Despite this, the feasibility study will not be carried out within the scope of the project. It is assumed that, since the project is being undertaken at next to no cost to Diagonal Solutions and the risks involved with creating the software are minimal, that, if such analysis was required, management would carry out the task before contracting the work to a third party.

If, however, the second activity of project management, planning, was not carried out the project would be doomed to failure from the outset. It is argued that the only way of building a full scenario of the project is to catch every fragment of detail as early as possible [2]. Clearly, this will be accomplished by considering carefully each aspect of the project before it is undertaken. However, Hughes and Cotterell state in [1] that “planning is an ongoing process of refinement” and that each iteration of planning becomes “more detailed and more accurate than the last”. As such information gathering will be required at the beginning of each stage to ensure that it is carried out effectively and on time.

Taking heed of this, the following Gantt chart (Figure 2.1) shows the break down of tasks with the planned execution dates, whilst the detail of these tasks will be discussed in later chapters.

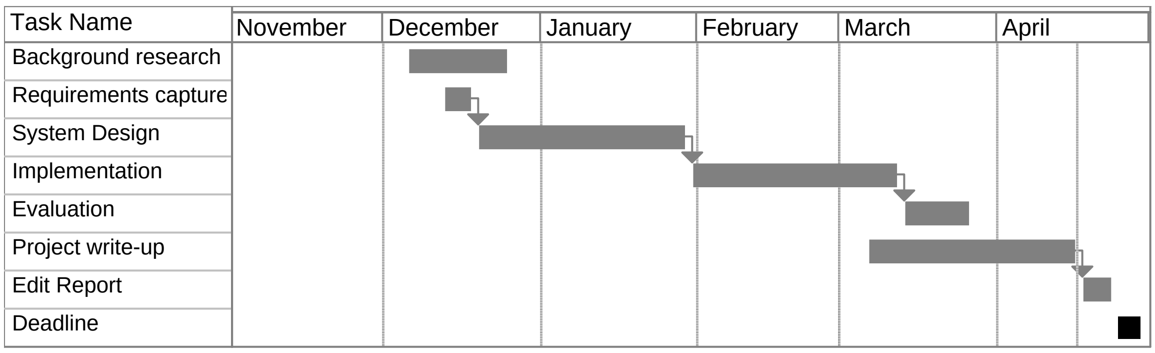


Figure 2.1: Project schedule

The plan does not take into account the work required in setting up the project with Diagonal Solutions before December. This requires choosing a suitable project and then ensuring that all the details are confirmed before any of the plans are made – much like the sale of any computer software.

PRINCE 2 is a “structured method for effective project management” developed in the late 1980’s by the Central Computer and Telecommunications Agency (CCTA) [3]. It provides managers with tools to carry out successful projects on time and within budget. Control over the project plans is very much a part of the PRINCE 2 framework. Controlling and revisiting the project plans ensure that the project:

“
- Is producing the required products which meet the defined Acceptance Criteria
- Is being carried out to schedule and in accordance with its resource and cost plane
- Remains viable against its Business case ” [3]

As mentioned above, the final activity involved in project management is the project execution. This stage often contains the design and implementation [1] stages which undoubtedly requires a satisfactory level of coordination. With a clear understanding of the projects activities, convenient communication channels are established in order to ensure that this coordination takes place. With all members of staff at Diagonal Solutions having access to email and their office being in Leeds, this is easily achieved.

There are a number of methods employed within the IT industry that can be adopted to achieve good project management [1][2][3]. Such methods are designed for large scale projects involving complex teams of developers and end users, and, as a result are not entirely relevant to this project. Although it is relatively small, it is hoped that by taking aspects of these frameworks and maintaining a dynamic approach throughout, a successful project will be achieved.

2.2 Development Methodologies

Having decided on a general approach to managing the project, a decision must be made on how best to execute the project plan.

Rapid Application Development

This methodology puts emphasis on quickly producing prototypes for the users to evaluate [1]. The methodology is not suitable for all projects, particularly ones that require thorough planning. It is best used when there is a focussed product scope, decisions can be made by few people, there are not many members in the project team, and the technical architecture is clear [4].

This would not be a suitable method for this particular project because, as yet, there is not a focussed product scope and the requirements are not clear. It is possible to adopt once all user analysis is complete, but then it will not constitute a methodology that can be applied to the project as a whole.

Joint Application Development

Joint Application Development (JAD) is a methodology where developers work intensively with users in workshops and agree documented business processes [1]. The advantage of using such a process is that all decision makers are generally in one room. This should speed up the communication process and enable decisions to be made instantly. On the other hand, the sessions may not cover everything that needs to be covered in one go. Prototyping, for example, could not take place unless many of these sessions were to be held.

Due to the nature of this project and of business itself, it is unlikely that JAD will be adopted as *the* methodology used. It is very unlikely that the directors, administration and support staff will have time to all sit down together for an extended period of time to discuss what, essentially, is a small project.

However, lessons can be learned from the methodology, and being based in Leeds, it is probably a good idea that meetings are arranged when the key decision makers will be around; this should help aid the communications process and ensure that decisions are made by the relevant people.

The Waterfall Model

This approach suggests making just one attempt at a project and getting it correct the first time [1]. “When it works well, the waterfall approach allows project completion times to be forecast with more confidence than with some more iterative approaches allowing projects to be controlled effectively” [1].

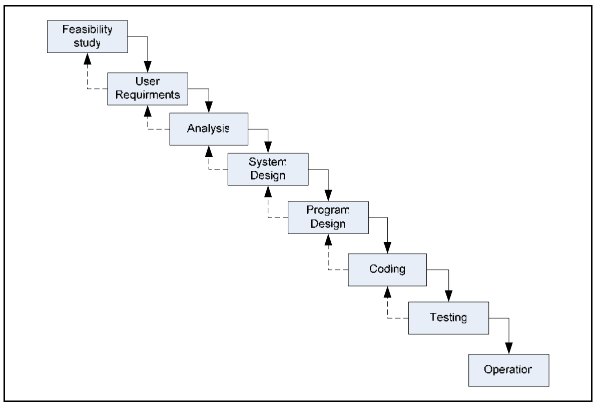


Figure 2.2: The waterfall model - adapted from Hughes and Cotterell [1]

Whilst the approach is generally suited to the project at Diagonal Solutions, it is felt that a prototype may be necessary to gauge users' initial feelings about interface design and the functionality of the system. The process has the advantage of being able to determine exactly which stage the project is up to. For this reason, the overall aspect of the Waterfall cycle will be adopted for this project, although it will be altered somewhat in the coding, and testing stages. Instead of adhering to the waterfall approach for these levels, a prototype approach should be adopted that will allow for iterative implementation techniques to be utilised.

Software Prototyping

Software prototyping enables developers to quickly portray the system functionality and as Hughes and Cotterell [1] mention, can be classified as either throw-away or evolutionary – that is, the prototype is either discarded after demonstrating it to the user or it is used in the next iteration of development. They also quote a number of reasons to use prototyping; a few are outlined here:

- Learn by doing
- Clarification of partially known requirements
- Production of expected results

Despite these advantages, there are a number of disadvantages that must be taken into consideration when using such an iterative process. Many of the steps required in good programming practice may be overlooked. Documentation, system testing and efficient programming may be overlooked if the final prototype is deemed acceptable to implement by the project champion [5]. These issues can be avoided by making the link between the waterfall model and prototyping methodologies. This ensures that the design is well documented and, as Diagonal Solutions are a small organisation, provides all parties with necessary feedback early on in the development cycle.

2.3 Summary

Many of the development methodologies and project management techniques above have been designed for large scale projects that require teams of developers working alongside numerous people with the client. In such a scenario it is clear to see that meticulous planning is essential to the success of the project. However, these may not form the best approach for a small project. By taking strengths from a wide selection of methods, the project should be made a success as long as care is taken throughout to ensure that the work is on schedule and that planning is communicated effectively with Diagonal Solutions.

The decision has been made to use marry the best features of the waterfall model with evolutionary prototyping to form the implementation and testing stages of the project. It is hoped that in doing so the both the advantages are gleaned from the use of prototyping but the disadvantages are avoided.

Chapter 3

Requirements Analysis

3.1 Business Analysis

Stakeholders

Before starting any analysis of the users' requirements for the proposed system, it is imperative that every stakeholder with a claim in the new system is identified to ensure that a full analysis can be carried out. Diagonal Solutions are a relatively small company that provide customised solutions in Information Management to a number of corporate and government bodies. The company employs approximately 60 employees who work both in Leeds and in remote locations throughout the UK. There are four key business units that operate in the company: administration, sales, development and infrastructure. The development team is split down into project teams based on the different technologies with which the company produces solutions. Because the sales department have no stake in the new helpdesk system, they will not be analysed any further.

The administration business unit is made up of four employees that are responsible for the day-to-day running of the business. Amongst other tasks, including accounting and contract management, these staff are responsible for the management of the support helpdesk. The development business unit makes up a large proportion of Diagonal Solutions' workforce and have the both writing software for clients and supporting these clients should any issues arise with their systems. The infrastructure team manage all of the internal systems and, where contracted, the systems of individual clients. They primarily deal with networking and software support issues and are very much support based as opposed to development. Heading the company is a team of directors though the only one with an input in the system is the Delivery Director who has overall control of the help desk.

From this high-level stakeholder analysis, it is clear that the sales team will have no interest in the new system, and as such will not be discussed any further. The remaining stakeholder groups can be grouped to form the following user groups:

- Helpdesk administrators
- Support technicians
- Directors

Further elaboration of these user groups will be necessary during the requirements analysis in order to gain a complete understanding of what the new system must achieve.

Customer Relationship

The analysis so far has only discussed the internal company but it is important to think of all stakeholders that have an interest in a system, and as such clients of Diagonal Solutions must be considered. In the present environment, customers that require support will call or email the help desk with their problems. This provides the customer with a single point of contact and ensures that resources are assigned according to the business impact to that customer (Figure 3.1). It is felt that ensuring the new system matches current working practices as opposed to bringing with it organisational change is a more satisfactory method than changing procedures and will lead to improved user relations that will ensure the customer interfaces with Diagonal Solutions in exactly the same method with the advantage that their request is dealt with more effectively.

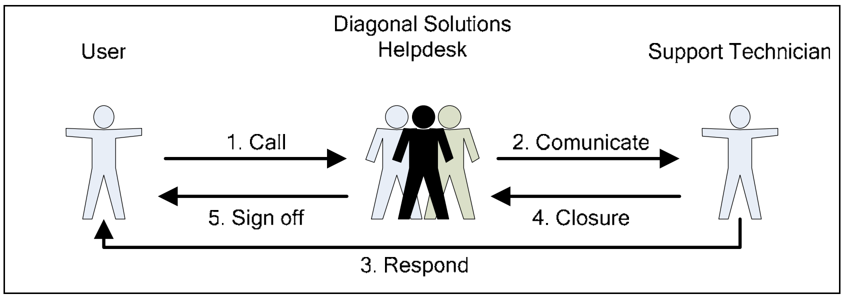


Figure 3.1: Helpdesk structure

3.2 Requirements Capture

Grasping a full understanding of the clients' requirements is fundamental to the success of the project. There are a number of approaches for obtaining the users requirements and each of these will be explored in order to acquire the broadest and deepest appreciation of what must be achieved in the system. The importance of requirements analysis isn't just for functionality. Without it, there would be no way of evaluating the system, as there would be nothing to evaluate against. There are five commonly regarded methods of gathering the requirements of a new information system that may be relevant to the project in hand.

Research

Many projects are undertaken by a third party that have limited understanding of the organisation they are about to carry out the project for. In many cases it is the first opportunity the analyst has to gauge an understanding of the business activities, processes and practices that may go on within the company [6]. There is no necessity for research or background reading into Diagonal Solutions because of the twelve months the author spent there as a support technician gaining understanding and experience of the procedures within the company.

Interviews

Though an understanding of the business is known, in-depth knowledge of the procedures many members of staff have to undertake is not apparent. Accordingly, interviews have been carried out with a number of personnel from different departments so that their requirements can be captured. Although they can be time consuming, and difficult to arrange, they have the capacity to be open-ended so that interesting facts can be explored when they emerge [6]. The results of the interviews (Appendix B) will be analysed in depth in the following sections. One difficulty experienced during interviewing was ensuring that the answers given were on track. Quite often it was found that interviewees moved off on tangents and wanted to discuss things that would relate to other systems and would not form part of this project.

Sample Documents

Bennett et. al. suggest the use of document sampling to "determine the information that is used by people in their work, and the inputs to and outputs from processes which they carry out, either manually or using an existing computer system" [6]. The sample documents collected from the interviews include a completed SLA form (Appendix D) and a document specifying the current call logging procedure (Appendix E). These will be particularly beneficial when designing the database as they outline exactly what data should be stored and some of the processed behind storing this information. However, it would be risky to rely solely on sample document. If the procedures are going to change then they are unlikely to form a reliable framework on which to base the new system but. They do however help to model the data structures and give valuable input to the design process

In addition to these sample documents, access has been provided to the current system so that its' implementation can be analysed and relevant processes abstracted. Care will be taken to ensure that the current system does not limit the scope of the new system. Whilst many of the features may need porting to the new system, they may also require substantial re-thinking of the processes going on behind them. As such, like the other documentation, the software shall only be used as a guide.

Questionnaires

It is felt that questionnaires are not suitable for this project. Although the system is in place for much of the company, by holding interviews with few people, but across the spectrum, interviews are regarded as being more efficient than a questionnaire. Although it is cited that it can be an economical way of gathering a lot of data, it is difficult to construct a good questionnaire [6]. There are too few people involved in the system to warrant spending time on creating a good questionnaire when this time would be better spent probing fewer people in interviews.

Observation

By watching people in their own environment it is possible for parts of their jobs to be uncovered that may not generally be brought out during an interview. It is also noted that people are not good at estimating the times it takes them to complete certain tasks [6]. Though it may be important to uncover as much detail as early on as possible [2] the prototyping methodology chosen for this project implementation was chosen in order to save on time consuming tasks such as observation. Whilst certain details may be neglected in early prototyping stages, these should be uncovered later on during system testing. As such it is felt that continuing to the design and implementation stages as quickly as possible will be beneficial to the project overall.

3.3 Requirements Analysis

The output from the requirements capture must be analysed to obtain a complete list of functional and non-functional requirements that can then be used to design and implement the system. This analysis forms a fundamental part of the system design stage in building on the requirements captured in the previous section and creating a comprehensive appreciation of what must be achieved.

Use Cases

One of the main uses of the interviews that were held at Diagonal Solutions has been to create *Use Case* diagrams that provide a high level model of what should be achieved in the system. A Use Case diagram provides not only a simple way of communicating ideas with users [7] but also, when complete, produces a high level understanding of what must be achieved in the system. The Use Cases were built up over a number of interviews with different people within the organisation including Mike Lord, the Delivery Director, Julie Jowett, the Business Administrator and a number of Developers and Support Technicians. These results were then summarised both to create the Use Case diagram but also to elaborate on each use case and generate a comprehensive requirements list.

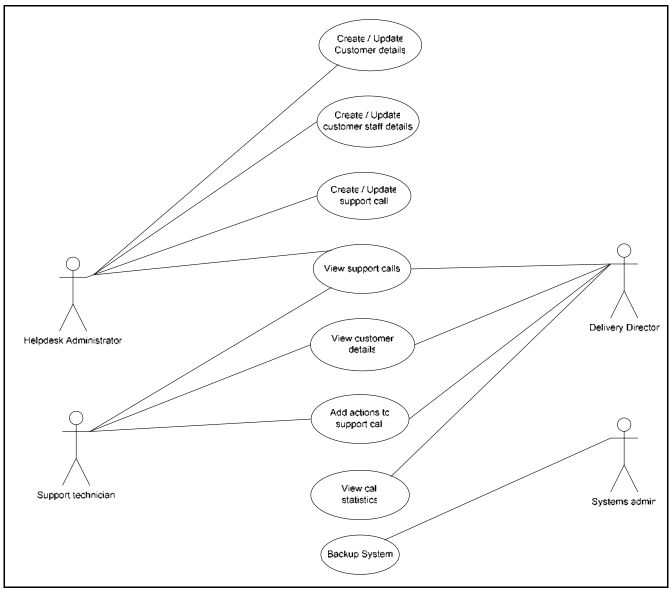


Figure 3.2: Use case diagram

Each of the Use cases in Figure 3.2 are described in Use Case Description forms found in Appendix C which discusses the functionality required in more depth, these should then, in turn, provide a better understanding of the system must achieve to aid the design process.

Functional Requirements

Based on the Use Case diagram above, a list of requirements is produced at a lower level of abstraction to that the design and implementation incorporate the appropriate functionality

One of the minimum requirements for the project is to produce "a suitable prototype call logging system with the ability to track cases and create reminders for support staff when deadlines approach in line with the clients Service Level Agreement". One question that must now be raised is what is deemed to be suitable for a prototype system? The decision taken in Chapter 2 to use hi-fidelity, evolutionary prototyping must be used, to some degree, as a guide. The following list of requirements outlines exactly what is considered necessary in the system, and whether or not the function forms one of the minimum requirements of the project.

1. Helpdesk Administrator

1.1. Manage support call

	Minimum Requirement?
1.1.1. Log a new call	Y
1.1.2. Update support call log Assign support call to a technician	Y

Close support call log	
1.1.3. Obtain status of call	Y

1.2. Manage Clients

1.2.1. Add customer to list	Y
1.2.2. Display all customers	Y
1.2.3. Display customer details	Y
1.2.4. Update customer details	Y
1.2.5. Enter SLA details	Y
1.2.6. Add / Remove client employee	Y

1.3. Manage System

1.3.1. Add / Remove system users	Y

1.4. Manage Contracts

1.4.1. View / Update default SLA response times	Y

2. Support Technician

2.1. Resolve customer issues

2.1.1. View assigned calls	Y
2.1.2. Update call log with action taken	Y
2.1.3. View customer contract details	Y
2.1.4. Pass call to third party	N

3. Delivery Director

3.1. Ensure contractual obligations are met

3.1.1. Chase technicians that have not updated an open call in a week	N
3.1.2. Beware of calls breaching / about to breach SLA	N
3.1.3. Raise priority of a call	N
3.1.4. Get statistics out of the system	N

4. Non-User Specific

4.1. Email Support call details to a system user	N
4.2. Email support technician when about to break SLA	Y

Table 3.1: Requirements list

Non-Functional Requirements

The non-functional requirements for the system form a set of the main desires of the users interviewed. It is wished that the system currently in place is improved upon in terms of the user interface. The security aspects must kept simple by relying on the user credentials entered when the user first logs in to Windows and, finally, the system must be easy to manage from a technical point of view. Though these will not be explicitly dealt with in terms of functional requirements, the design must be based around these concepts – they are no less important than the functional requirements.

3.4 Best Practice Guidelines

The IT Infrastructure Library (ITIL) is a set of best practice guidelines that has been widely accepted as the industry *de facto standard*. The ITIL standards break down service requests into several distinct areas:

- Incident Management
- Problem Management
- Configuration Management
- Change Management
- Release management

Understanding how the system should incorporate these areas is important before the design stage is entered. This section concentrates on the relevance of each of these areas and considers their inclusion in the new system. Whilst these are arrived at from best practice guidelines, they may not be appropriate for the kind of support that Diagonal Solutions provide or may be out of the scope of what the system should deal with.

Incident Management

The CCTA state in [8] that “the primary goal of the Incident Management process is to restore normal service operation as quickly as possible and minimise the adverse impacts on business” which must be considered to be the main function of the helpdesk system because it is a fundamental role within IT Support that incidents are managed effectively. When not appropriately dealt with, incidents will lead to extended periods of decreased productivity or none whatsoever. The CCTA also state that “normal operation” should be defined within the service level agreement [8]. Whilst this may be done at Diagonal Solutions, there is nothing in their current system to ensure that the service level agreement is met. Functionality in the new system support would ensure that the clients’ SLA agreements are adhered to and that penalties imposed upon failing to reach these agreements are minimised. The requirements capture has stated that this must be in the new system and, as such, forms part of the minimum requirements of the project.

Problem Management

Going a stage beyond incident management leads towards an enriched system capable of managing problems – the underlying causes of many incidents. “The goal of Problem Management is to minimise the adverse impact of Incidents and Problems on the business that are caused by errors within the IT Infrastructure, and to prevent recurrence of Incidents related to these errors” [8]. By incorporating problem management into the new system the project would be made much more complex. It is felt that this area should be kept outside the scope of the project for several reasons.

Firstly, the number of calls that Diagonal Solutions receive is relatively few when compared with many corporate helpdesks. Additionally, with few technicians communicating problems between them very effectively by email this is not currently a priority. Whilst it is felt that having some form of problem management mechanisms within the system, it is hoped that a well designed database will easily support future application enhancements without fundamental system changes.

Configuration Management

Central to much of the support that must be provided to customers is the configuration of their network, servers and personal workstations. The customers who outsource all of their IT to Diagonal Solutions may expect some form of management of their assets. The information that is held about the customers systems is held in a Microsoft Word document, providing instructions on how to remote control their systems and on what is installed on their servers and workstations. Although it may be beneficial to hold this information together with the incident log (for example, to provide a sound basis for support of an incident [8] or to identify the computer that a certain user has allocated to them) it expands the scope of this project too far and would push timescales significantly. This is something that could, and probably should be considered and implemented in future versions of the helpdesk system but, because the documentation is already held elsewhere, it does not form part of the minimum functional requirements for the successful day-to-day management of the helpdesk process.

Change Management

As described in the ITIL guidelines “change arises as a result of Problems, but many Changes can come from proactively seeking business benefits such as reducing costs or improving services [8].” It is felt that change management should have processes that remain separate from the day-to-day running of the helpdesk. Whilst the need for certain changes may arise from an underlying system problem with, it is more than likely that any change would be managed by an account manager, not by a support technician. Implementing this kind of functionality may increase the complexity of the system unnecessarily because current systems (be them software or human) already deal effectively with this process.

Release Management

The CCTA dedicate a chapter to release management in [8], however, it does not deal primarily with support issues, or more precisely, problem issues. Whilst it may be important to the entire support process to manage releases of hardware and software to the customer, the aim of the system is to provide functionality that aids the technician when faced with a new support call, not when undertaking a new project for a customer, which, in itself will have a different set of procedures to adhere to.

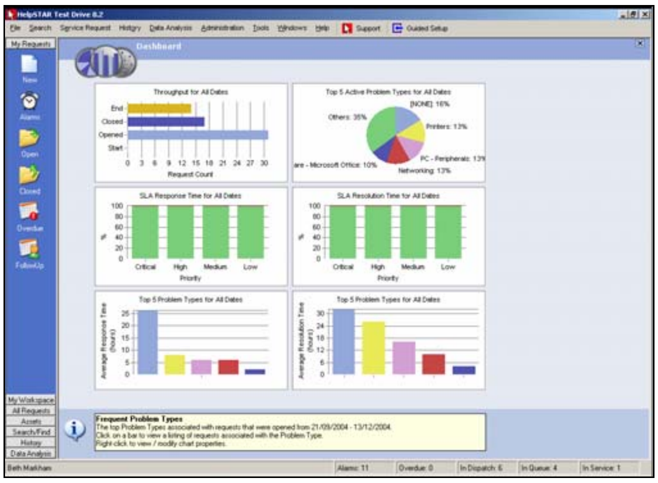
Chapter 4

Helpdesk Software Market

From the analysis undertaken in the previous chapter it is felt that Diagonal Solutions' requirements are quite unique. Whilst many businesses are likely to operate an IT helpdesk, they are unlikely to require the functionality needed within Diagonal Solutions. A comparison that could be made is with the helpdesk at Diagonal Solutions and the *Information Systems Services (ISS)* helpdesk operated by the *University of Leeds*. The two institutions are very different indeed. Whilst the ISS manage all of the systems for one body, split into numerous departments, Diagonal Solutions manage systems of many different organisations, each with their own contract. Identified in this section are a number of solutions aimed at corporate helpdesks with the similar objective of Diagonal Solutions – to provide the user with a customer focussed support environment that facilitates the successful management of individual system issues. Whilst it may be interesting to focus on the technological aspects of the each system, analysis will be carried out, and a decision made about the best approach to be taken for the project based upon the benefits the software brings to the company not the technology it utilises.

4.1 HelpSTAR 8.2

HelpSTAR is an out-of-the-box package aimed at the mid-market and has the functionality that each of Diagonal Solutions' individual clients are likely to require. One particularly nice feature of the system is its business workflow rules that allow an administrator to control requests from the users to the correct people in the organisation. It also takes parts of the ITIL best practices and as a result rich functionality is provided through the inclusion of problem management and asset management components. Additionally it meets Diagonal Solutions' requirement to escalate the priority of a call and to generate alarms for when this is done. The *Advanced* and *Enterprise* versions of the software are also packed with a web portal and comprehensive reporting solutions that are both useful to the Diagonal Management and to their clients. In spite of these well-packed features, in a very nicely presented user interface, there is one major problem with this software – it has absolutely no support for setting up different clients and, as such, would make charging customers for work more than a headache. In fact, charging for work would not be the first problem. It would be completely impossible to store details for each of the customer and their employees as the package is only designed to be used on the helpdesk of an individual organisation. Installing it for each organisation is not a possibility either, both down to the practicalities of this and the fact that it would not be possible to track calls efficiently from one central location



Frequent Problem Types
The top Problem Types associated with requests that were opened from 21/09/2004 - 13/12/2004.
Click on a bar to view a listing of requests associated with the Problem Type.
Right-click to view / modify chart properties.

Figure 4.1: Screenshot of the HelpSTAR Dashboard (System Statistics)

4.2 Intuit Track-It! 6.5

Like HelpSTAR, Track-It! has a well presented user interface that, without performing a comprehensive Human-Computer Interaction analysis, seems to lead to good usability. Many of the features are similar to those in HelpSTAR; it is hard to differentiate the two products. Again, this product has no facility for creating numerous customers and will not support the requirements of Diagonal Solutions. However, a feature from both this and the previous package could be nicely adapted into the new system. Each system provides a statistical analysis screen (Figure 4.1) that provides the user with real-time information about the support calls that are open on the helpdesk. It outlines how many calls are within and in breach of internally configured service level agreements and displays this information through the use of graphs.

4.3 Bugzilla

Bugzilla is currently used within Diagonal Solutions for bug tracking in the software that they produce for clients. It is an open source package from the makers of the Mozilla web browser [9], that, when installed is very bare. It therefore requires a lot of configuration and personalisation before it can be used effectively. As the package stands, it is not suitable for issue tracking of support calls like the previous two packages. It is very much designed around the tracking of bugs within software packages as opposed to tracking of issues with users' computer systems (for example, it does not have a field for the type of call such as 'hardware' or 'software'). In order to get the system running as a suitably featured issue tracking system, it would need a huge integration project to be undertaken in order for this to happen and even then, it may not be as good a solution as the previous two packages. However, the system has several advantages. It is web based and as such can be accessed from anywhere within the organisation, without the need for software to be installed on the clients PC. Also, because it is open source it is possible to adapt the system to the companies' requirements. However, the work required to do this is likely to outweigh the work required to create an entirely new system from scratch. It is often harder to look at code and determine what it is doing than to sit down, design and implement a package from scratch.

4.4 Summary

The market research has shown that no solution provides the functionality for setting up different organisations but, at best, only allows for different departments to be configured within the company. Whilst this sort of 'off the shelf' approach may be acceptable for a helpdesk that serves just one organisation, it is far from ideal for an organisation that has to serve many separate businesses with numerous contractual arrangements. Although it may be possible to track a package down that has the functionality required, the costs involved in performing the research and in procuring software that may still require changes, or may be too sophisticated for Diagonal Solutions' needs, are more than likely going to outweigh the costs of implementing a system in house.

Chapter 5

Technical Analysis

5.1 User Interface

The issue of usability is an important issue that must be addressed during the design of the user interface. Though the system is not intensively used, it is still important to try to reduce the time it takes to carry out a task. While it is perceivable that cutting the time it takes to carry out a task by fractions of a second may save certain companies, employing hundreds of people carrying out the same task, a lot of money, it simply wont be the case at Diagonal Solutions. More importantly for the company is the need for the interface to be designed so that learning the system is simple and carrying out day-to-day tasks is more efficient than the current system. Many web based systems fail to combat basic usability issues and end up being less efficient than their predecessors.

It is believed that by looking at current methods and theories in Human-Computer Interaction (HCI) and by evaluating the user interface on the current system, a more effective interface can be achieved in the new system.

Getting HCI issues solved in systems is not an easy task to carry out and requires knowledge from many fields of science. As Dix et al discuss in [10] "the ideal designer of an interactive system would have expertise in a range of topics: psychology and cognitive science to give her knowledge of the user's perceptual, cognitive and problem-solving skills; ergonomics for the user's physical capabilities; sociology to help her understand the wider context of the interaction; computer science and engineering to be able to build the necessary technology; business to be able to market it; graphic design to produce an effective interface presentation". Clearly, an Information Systems degree programme is not going to cover every area that is preferred in HCI but it is possible to pay careful attention to the design of the system to increase, as much as possible, the usability to its users. This will be the focus of this section.

Task Identification

Shneiderman and Plaisant offer a number of principles in interface design [11]. They discuss the need to identify and think carefully about the tasks that the user must carry out. This has been achieved in the requirement analysis above. However, Shneiderman and Plaisant go further by monitoring the frequency at which users carry out the tasks. They also explain how tasks with a high frequency will benefit by assigning shortcut keys to them thus aiding the user in quick navigation. Although no formal frequency observation has been carried out, it is clear, due to the limited number of tasks they carry out, that the administrators' task of managing a support call will be most frequent, with client, contract and system management being used much less frequently. In terms of the support technicians and delivery director, because they only have one high-level task to carry out, this form of analysis is not relevant and breaking these functions into smaller tasks is not feasible beyond those tasks discussed in the requirements analysis.

Short Term Memory

The short term memory of a human is used to remember details to carry out every day tasks. For example, a simple multiplication may be divided up into separate blocks, each block requiring its short term storage in memory. The capacity of this memory is very limited and people can usually remember 7 ± 2 facts [10]. It is important to note that a system should not rely on the user's short term memory very much at all. For example, they should not have to remember information from one page to complete the next. However, it is important that a page does not become so cluttered with information that the user cannot possibly store enough of the information in short term memory to be able to realistically complete their task.

Execution-Evaluation Cycle

This is a model of user behaviour designed around the execution of a command on a computer system. If the user moves the mouse, the system then moves the cursor on the screen, and the user then evaluates the response the system had to their command [10]. There are several stages in this process which starts off by the user establishing their goal. The user then forms an intention to carry out actions and then decides on the actual actions they are going to take. The user then executes those actions and perceives the changes they have on the system. They then interpret and evaluate the system state after the action has been carried out [12]. This shows that user inputs should be intuitive and that the outputs should not give unexpected results. It is important to ensure that the system adheres to common standards that users have grown to expect and to ensure that it does behave in a manner that the particular users at Diagonal Solutions expect.

Form Filling

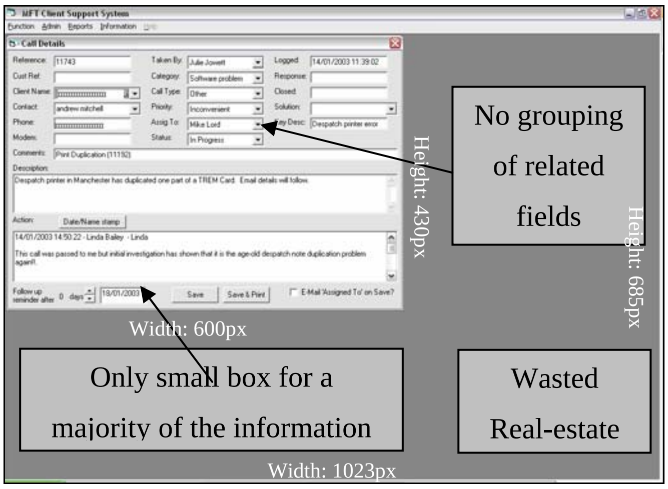
One area of the system where usability will be fundamentally important is the data entry of new support calls. Often the calls will come in on the phone and so the page must be easy to navigate and to enter data into, possibly using just one hand. The literature read on this subject has not elaborated on how one should make a form usable. Dix et al state that "most form filling interfaces allow easy movement around the form and allow some files to be left blank. They also require correction facilities, as users may change their minds or make a mistake about the value that belongs in each field" [10]. However, we must go further if we are to pay real attention to the data entry form. As mentioned, the form must allow for easy movement around it. In theory it should be as easy as moving a pen to the relevant area on a paper form. Though this may not be possible in practice, by simply ensuring that the `'Tab'` key results in the cursor being moved to the next field and not some random space is one step towards a usable system.

Menu's

A system that has many different parts will, by definition, require some form of navigation to those parts. Because menus rely on recognition, rather than recall, it is important to ensure that they are meaningful and by clicking on them have the desired effect [10]. Menus can be hierarchical and as such grouping of tasks can be done. This aids the navigation process and an important factor in the helpdesk system will be to ensure that, where applicable, suitable measures are taken to make hierarchical menu structures effective rather than hinder the user.

Current System

The current application was first developed in Visual Basic using Windows forms but little or no attention has been paid to any Human-Computer Interaction (HCI) issues. The layout and positioning of fields is poor with little consideration being given to the physical size of the field and the amount of data that it should store. Furthermore, though the application can be maximised to utilise all of the screen real-estate, the form for the help desk functions does not (Figure 5.1). This leads to extremely poor use of the screen and as a result poor usability. In fact, the dark grey that has absolutely no use accounts for 64.2% of the screen – a sheer waste of valuable space that could be used to display more of the support call details or to improve the actual layout of fields.



Reference:	11743	Taken By:	Julie Jowett	Logged:	14/01/2003 11:39:02
Cust Ref:		Category:	Software problem	Response:	
Client Name:	[redacted]	Call Type:	Other	Closed:	
Contact:	andrew mitchell	Priority:	Inconvenient	Solution:	
Phone:	[redacted]	Assig To:	Mike Lord	Key Desc:	Despatch printer error
Modem:		Status:	In Progress		

Comments: Print Duplication (11152)

Description: Despatch printer in Manchester has duplicated one part of a TREM Card. Email details will follow.

Action:
14/01/2003 14:50:22 - Linda Bailey - Linda
This call was passed to me but initial investigation has shown that it is the age-old despatch note duplication problem again!!.

Follow up reminder after 0 days 18/01/2003

E-Mail 'Assigned To' on Save?

Figure 5.1: Current user interface

The Interface is also somewhat outdated in terms of the information that is displayed or captured when logging a new call. “The primary goal for … [form-filling], and dialog-box designers is to create a sensible, comprehensible, memorable, and convenient organization relevant to the user’s tasks” [11]. By displaying fields that are no longer required (or understood) users find themselves confused. For example, the ‘Modem’ field was originally designed to be used to store the telephone number to dial into the customers system. None of Diagonal Solutions customers use this method of remote access anymore and as such, the field is redundant. These issues must all be addressed in the design of the new system; however, good points must also be taken from the current design.

Summary

As has been proven by historical events such as the London Ambulance Service Computer Aided Despatch project [13] usability is not a ‘nicety’ or an ‘added-bonus’ – it is critical to the success of any computer system. Though not in a life-critical environment like the London Ambulance Service, by paying attention to the usability issues with the new system, the users experience of the system should be a good one and their effectiveness should increase as a result of this.

5.2 Diagonal Solutions Infrastructure

Diagonal Solutions run a mainly Microsoft based server infrastructure with just a couple of Linux servers utilised for support certain customers. The business infrastructure is broadly supported by:

- Microsoft Active Directory 2000
- Microsoft Exchange Server 2003

In addition to these, there are a number of servers that host the Intranet and development projects. The servers can all be identified as characters in J.R.R. Tolkien’s ‘*Lord of the Rings*’. This naming convention will be adhered to throughout to try and ease the description of each server role. Figure 5.2 below is a simplified model of the, more complex, internal network (Appendix G) but it does show the servers that must be analysed.

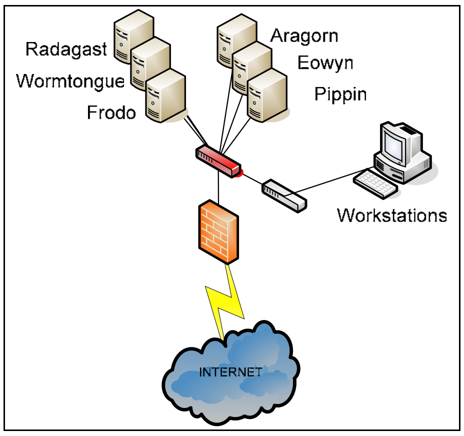


Figure 5.2: Simplified network diagram

Microsoft Active Directory 2000

Served by: Radagast and Pippin

Microsoft Active Directory (AD) is a Directory Service based on the open Lightweight Directory Access Protocol (LDAP) standard. The LDAP directory service is a system in which the organisational hierarchy can be stored and attributes of the users such as their password and contact details are managed [14]. Because AD is based on open standards it is interoperable with other directory services that employ the same standard [15]. More importantly, there are several application programming interfaces (APIs) [15] that enable easy integration of LDAP directories into any software application. All of the employees at Diagonal Solutions have their own AD entry configured with their password and contact details. Given an appropriate choice of development platform integration of AD in the new system will be possible.

Microsoft Exchange 2003

Served by: Wormtongue

Microsoft Exchange server is an email server with which Microsoft's email client, Outlook, is designed to cooperate with. Each user has their own mailbox on the server that they connect to every time they open Outlook. No email is stored on the clients' computer unless they specify local caching of email. Like AD, Exchange enables the use of standard protocols such as SMTP and POP3. In addition, Exchange integrates with AD and as a result enables central management of users' mailboxes as well as easy lookup of email addresses within the organisation.

Microsoft Internet Information Services

Served by: Eowyn

Internet Information Services (IIS) is a component of the Microsoft Windows Server operating system that, given the appropriate infrastructure, provides web hosting capabilities to the machine it is installed on. IIS provides support for Active Server Pages (ASP) and ASP.NET applications. These are proprietary web development platforms discussed later. Like many other Microsoft products, IIS integrates completely within the AD infrastructure but can also be used as a standalone server that manages its own security.

Microsoft SQL Server 2000

Served by: Aragorn

There are many different relational database management systems (RDBMSs) from a number of software manufacturers both from the open source and proprietary markets. Diagonal Solutions' decision to use Microsoft SQL Server 2000 is consistent with the choices made with the rest of the infrastructure. All of the products marketed by Diagonal Solutions use this product so the knowledge within the company is high. However, their decision to use this software may have impacts on the design of the helpdesk system. It is important to know the limitations of using Microsoft's SQL server product as opposed to others such as Oracle or MySQL. These are discussed in section 5.3.

Desktop Applications

Every member of staff at Diagonal Solutions has a personal computer or laptop with a standard configuration. Though some staff may require additional software the core components will always be installed (see 0). Knowing exactly what the clients configuration is makes implementation of a system much easier. For example, web developers often have difficulty in ensuring their pages are compatible with a range of web browsers (Netscape and Internet Explorer being prime examples). Because it is known that all the clients will be running Internet Explorer 6 designing the page will be much easier.

Summary

Throughout this section many of the Microsoft technologies have been discussed. Because Diagonal Solutions are a gold partner of Microsoft they are entitled to dramatically discounted software and as a result, most of their infrastructure is based on this. It is felt that discussion of other technologies such as Oracle, or UNIX based systems would be inappropriate simply because the implementation of such systems would not be entertained by the management at Diagonal Solutions. Skills within the company with other technologies are highly limited and there would be little or no support available for other technologies. It is important, however, that whatever technology is chosen for the implementation of the system, it is compatible with the current infrastructure and that significant leverage from this technology is achieved.

5.3 Technology Options

Having now discussed the environment within which the new system must work, it is possible to examine the different development technologies available.

Client / Server Architecture

The infrastructure at Diagonal Solutions provides an environment that allows for a client / server architecture. The decision that must be made is whether to implement a thin-client or fat-client architecture. A thin-client architecture ensures that all of the business logic is controlled and the information is received from the server is in its final state [16] and as such has no further processing to do before it displays the information. A good example of a thin-client architecture is that of a webpage where processing may be done on the server to obtain data to be displayed, but the client receives only the information in its final state – the HTML. In contrast, with a fat-client, much of the processing of data is done on the client's computer and as such, requires more powerful machines. However, the latter results in much of the load being taken away from the server [16]. There are many examples of fat-client applications, but perhaps the most relevant example is the helpdesk system currently employed by Diagonal Solutions. This is a Windows application that must be installed onto each of the individual machines that require access to the system.

There are a number of issues encountered with this architecture. Installation of the application is non-trivial; it takes approximately ten minutes to install onto each machine and requires manual ODBC (discussed later) configuration. Any updates made to the application also result in a requirement to redeploy throughout the company – far from ideal. Whilst the current architecture takes load away from the server, much more data will be transferred between the client and the server than with a thin-client. Consequently, it is felt that moving to a thin-client architecture would lead to a number of benefits that must be exploited in the new system.

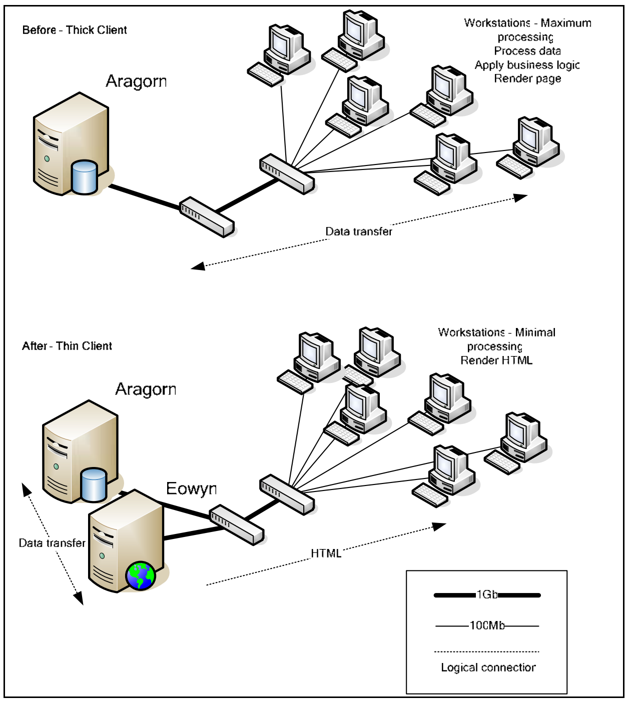


Figure 5.3: Current and proposed architectures

All the servers within the company are connected via a Gigabit network; workstations only benefit from a tenth of this speed. In recognition of this it is advantageous to have most data transferred between the servers and not the workstations (Figure 5.3). By transferring only the presentation of the application, the bandwidth restrictions between the server and the workstation are less of an issue. It is also useful to be aware that the power provided by a dedicated server is more appropriate use of processor power than relying on a PC tasked with doing a number of jobs.

This leads to two options. Either the new helpdesk system is developed using web based technology or an architecture is sought after through the deployment of technology such as the Citrix MetaFrame Server found at the University of Leeds, School of Computing. As Diagonal Solutions do not have the infrastructure in place to support this style of thin-client architecture, a client-server approach will be adopted in the form of a locally hosted Intranet site using the existing infrastructure.

Web Technologies

Since the birth of the World Wide Web technology has moved on considerably from static HTML pages to dynamic, data-driven solutions that have the capacity to support entire business processes. Having an understanding of the technologies available and their limitations is fundamental to the correct implementation of the help desk system. Three technologies are to be discussed: PHP, Active Server Pages (ASP) and ASP.NET. It must be decided if, and how, each of them would fit into the infrastructure at Diagonal Solutions and an understanding of how the technology would integrate with the requirements outlined above must be gained.

PHP

PHP is a server-side scripting language that provides database access, form processing, user authentication and many other tasks [17]. In short, it has the capabilities for developing the help desk system with all of the functionality described in the requirements' analysis. Often it is used in conjunction with the Apache web server that is run on approximately 69% [18] of the Worlds Web servers. It will also run on both Microsoft Windows and on the Linux platform.

However, Diagonal Solutions do not use Apache and have IIS enabled on their internal facing Intranet server. It is possible to install PHP on an IIS installation but no-one in the company would be able to support this as well as the Microsoft technologies they have already adopted. Despite the fact the PHP would provide all the functionality required in the new system, it would be unwise to endorse its use in this environment. "Trying to incorporate existing Microsoft application data into a PHP run Web site would require starting from scratch with a great deal of headaches including purchasing new programs" [19]. This, coupled with the issue of no direct support for Windows, leads to an alternative that offers similar functionality but integrates more transparently into the infrastructure to be sought after.

Active Server Pages

Much like PHP, Active Server Pages (ASP) offers a server side scripting environment but it is possible to write pages using a number of different programming languages [20]. Again, like PHP, ASP development provides a server-side scripting environment in which pages are rendered 'on-the-fly' to return dynamic, data-driven content in an HTML page. Moreover, it is technology that aligns perfectly with the current infrastructure and employee skill-set at Diagonal Solutions.

On the face of it, ASP appears to be a suitable environment for the helpdesk system. However, Microsoft's latest web development tool, ASP.NET maybe a more appropriate and advanced technology for the application development.

ASP.NET

Before discussing the use of ASP.NET a short introduction to Microsoft's .NET framework should provide the reader with a better understanding behind the technology that supports it. Grimer describes the .NET framework as "An environment in which there is a common transparent foundation layer through which any programming language can access either data or operating system functionality" [21]. Two key parts of the framework are the .NET class library and the Common Language Runtime (CLR). The CLR enables the framework to interpret a number of different languages supported by .NET. Regardless of the language used (VB.NET, C# or a number of others) the source code is interpreted by the CLR into a lower level managed code, Microsoft Intermediate Language (MSIL), that is then executed within the .NET environment [21].

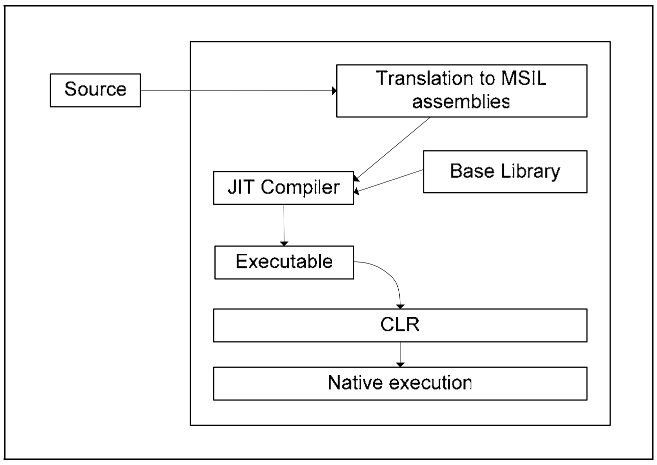


Figure 5.4: The execution model for the .NET architecture – adapted from [22]

The architecture of ASP.NET leads to a flexible environment capable of coping with a number of languages if necessary. One of the more important features of ASP.NET is that its' architecture separates the HTML mark-up from the server-side scripts. This enables the layout of the page to be separated from the functionality. It is felt that this is conceptually a simpler model than the old ASP method of inserting server-side scripts within the HTML. Another advantage of this architecture is that the use of the CLR results in an object-orientated solution. It goes far beyond the server-side scripting model of ASP by implementing each page as a class. Each page inherits from the class `Page` which defines a set of minimum methods and properties available to all pages [22]. This object-orientated approach not only allows for the code of each page to be stored separately from the HTML but also enables cross-functionality between pages through the utilisation of class inheritance.

Data Access

One of the fundamental processing requirements of the new system is that of the storage of data. Some connectivity between the chosen development technology and the underlying Microsoft SQL Database is required. Some discussion of these database connectivity methods is required before a suitable application design is established.

Open Database Connectivity (ODBC)

"ODBC is a C programming-language library that uses SQL to access data" [23] and provides a good interface into a relational database [24] that supports it. This hides the actual choice of database from the developer but provides a standardised method of accessing its data. This is the method that Diagonal Solutions use in their current system. However, the technology is aging and, reflecting on the decision to develop the system using ASP.NET, a more contemporary method may provide enhanced functionality and improve the implementation.

ActiveX Data Objects (ADO)

Unlike ODBC, ADO provides uniform access to many more data sources than relational databases [23] such as email and multimedia files. It is an extension to ASP and supports many more features than simple data collection and retrieval [24]. One of the key features that developers at Diagonal Solutions exploit regularly is *Stored Procedures*. As the name suggests, a stored procedure contains a SQL command that can be called by the application programmer at any time. The procedure can be designed to accept and return different variables providing a reasonable level of flexibility. This also takes some load off the network because the SQL code is processed on the server and so fewer instructions are passed between the client and the server [24]. It also means that if a query is needed regularly, the stored procedure can be called without the need to remember any complex SQL.

ADO.NET

With the launch of the .Net framework from Microsoft, ADO.NET was presented as a method for .Net applications to connect to databases. It addresses the problems that were encountered with ADO and the disconnected data access model necessary in a Web application [24]. Using ASP and ADO requires the developer to ensure that the database connection is opened and closed appropriately to avoid runtime errors. This complexity is avoided in ASP.NET by providing a two-layered model that consists of a *connected layer* and d*isconnected layer*. The connected layer offers similar functionality to ADO that delivers the data to a number of objects in the disconnected layer. Data can then be manipulated in a *DataSet* without the need for a continued connection to the actual database [24].

Summary

Throughout this section a number of options have been considered and the decision to embrace the Microsoft .NET framework has been taken. Although this is the preferred method of Diagonal Solutions, because of their expertise in the area and their ability to support it, the decision has been taken on the merits of the technology itself. ASP.NET coupled with ADO.NET provides the most advanced functionality with better flexibility from the alternative approaches discussed. It addresses issues that have been uncovered historically and also integrates into the infrastructure that Diagonal Solutions' core business relies on.

The choice of technology has not been a trivial one. It is accepted that whichever tool was chosen there would be somewhat of a learning curve during the implementation. However, whilst the author has experienced both ASP and PHP, very little experience of ASP.NET has been encountered. It would, however, be naïve to think that the areas of experience will help with this project. It is clear that whichever method chosen would result in a learning curve but the choice has been made because of the qualities the technology brings to the project not because of an understanding of syntax – this can be easily learned.

Chapter 6

System Design

It has already been decided that an evolutionary prototyping approach is to be taken, so why concentrate on the design of the system? The answer to this is simple, without a suitable system design the issues discussed in Section 2.2 will not be overcome. Whilst the prototyping method is considered the best approach for this project, it is also felt that documentation and efficient programming practices must be realised. As a result, the analysis from the previous chapter will aid the design of the system before any implementation is to take place.

6.1 Application Architecture

The architecture of the application is designed to fit into the infrastructure at Diagonal Solutions which leads to a distributed n-tier application. This is defined as a model that helps developers to create flexible and reusable applications by breaking it up into different tiers [25]. As a result, it is likely that if changes are made to a single tier, it is possible that the entire application will not need updating when those changes are made.

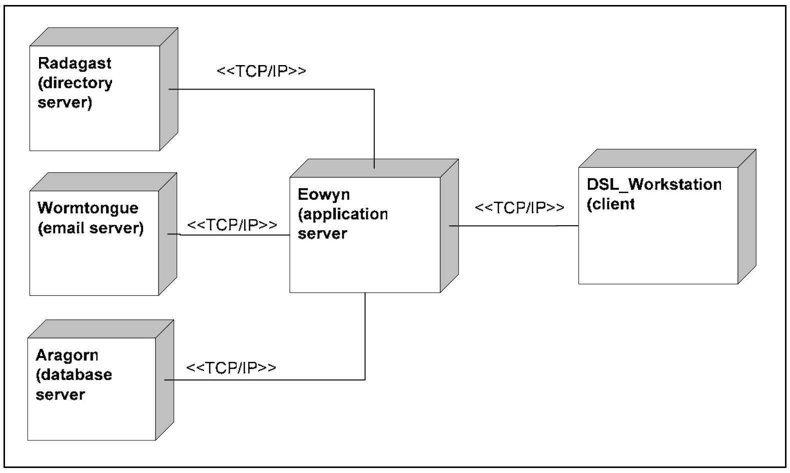


Figure 6.1: Application Architecture - UML Deployment diagram

Taking the n-tier path also leads to a number of benefits created by the integration of the internal infrastructure. The helpdesk application itself will not deal with the security aspects – Active Directory will handle this. The solution will use the current SQL Server and will utilise email facilities provided by the Exchange server (see Figure 6.1 for the architecture summary or Appendix M for entire architecture). Finally, the client communicates with just one server, Eowyn, which brings together each tier in a single point of entry for the client and references other tiers when they are required.

Though the system architecture is reasonably complex it leads to a number of benefits for Diagonal Solutions that will simplify the management of the new system. Because IIS will not have to deal with security, password management remains centralised in the organisation, and a single logon is retained. By including a tier for dealing with email the system will be capable of fulfilling one of the minimum requirements of sending email when a call is close to breaking its SLA. Finally, by keeping data separate from the application it provides a much more flexible framework. The application could be easily redesigned without having to redesign the data structures. Likewise, in theory at least, the data storage mechanism could be altered without having to make changes to the actual application. In addition, unless the application itself changes, significant improvements could be made to the implementation in the future without the users of the system being aware of the details of these changes (for example, that database could be moved to a faster server; users would not know the details of the change apart from the fact that their queries had been executed in less time).

6.2 Database Modelling and Design

A sound database design will be the key to the success of this system. By ensuring that, at this stage, all the relevant information is incorporated into the system, the rest of the system design will fall easily around the database. Database design is generally thought to involve the modelling of different *Entities, Relationships and Attributes*. In [24] Connolly and Begg break down the design of the database is broken down into different stages:

- Conceptual database design
- Logical database design
- Physical database design

The conceptual design stage is used to build an understanding of each of the entities, relationships and attributes that have been identified. This is then translated to form a logical design by creating valid relations. The physical design must then be created and will be dependant on the Database Management System in use [24].

Conceptual Database Design

The Entity-Relationship (ER) diagram allows the database designer to get a clear picture of how different entities relate to one-another. “Requirements analysis is the most important step (step I) of the database life cycle and is typically the most [labour] intensive” [26] and as such the previous analysis is extremely important. In order to determine the relevant entities, relationships and attributes, the internal documentation is the most useful (see Appendix D and Appendix E). These documents outline the entire call logging process and also give an insight into the data that has to be captured with each call.

The E-R diagram in Appendix J illustrates the entire conceptual database design. This is summarised below by displaying just the entities and their relationships. However, this design was not conceived in just one attempt. Each time an E-R diagram is completed it is imperative that it is validated to ensure no traps have been fallen into.

In the first draft on the E-R diagram (Appendix I) many errors were identified between the entities 'Customer', 'Helpdesk' and 'Support Call'. The relationship 'agrees' comes from the entity relationship between 'Customer' and 'Service Level Agreement' but has no meaning between the 'Customer' entity and 'Helpdesk' entity. Ignoring the 'Service Level Agreement' entity, the relationship between 'Customer' and 'Helpdesk' would be calls. However, as identified in Figure 2.1, just doing this causes in a connection trap. Though the relation would be read "A customer calls the helpdesk who logs a support call" there is no way of identifying which support call belongs to which customer.

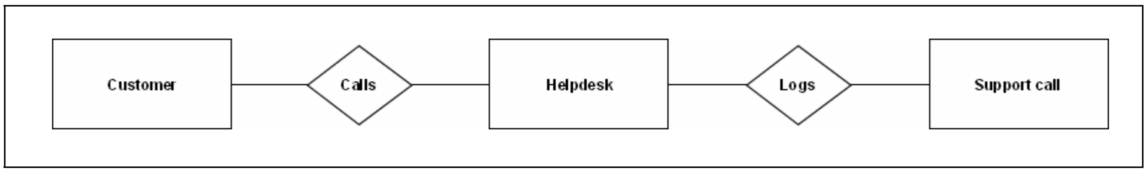


Figure 6.2: Customer, Helpdesk and Support calls relationships

To overcome this, the 'Customer' attribute must not relate to the helpdesk; after all it is a member of staff of the customer that will make the call not the customer itself.

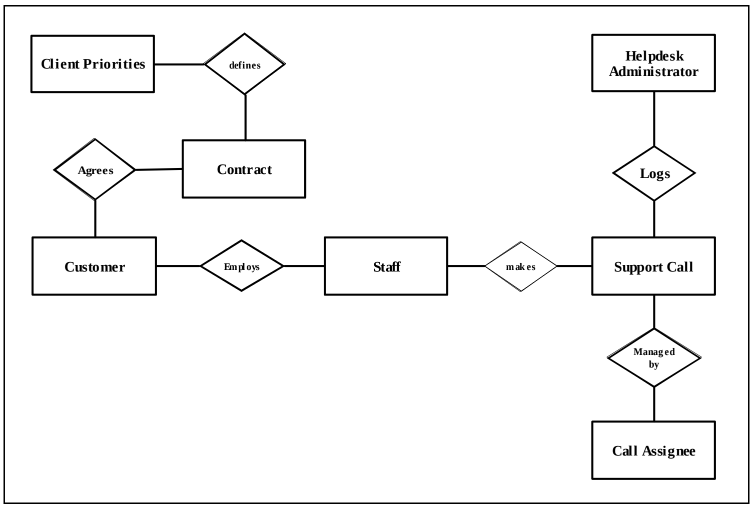


Figure 6.3: Summary E-R diagram

The example above outlines the importance of getting the design correct. If this task had not been carried out and a database had been implemented without due care and attention, it is likely that during the implementation of the web front-end issues would be encountered or at the very least, writing valid SQL queries would much more complex.

Logical Database Design

With a conceptual model completed, the logical database design which incorporates the tasks of deriving relations, validating the relations using normalisation and ensuring that all constraints are checked [24]. From the conceptual model (the E-R diagram) Connolly and Begg state that the following must be carried out:

- For each strong entity in the model, create a relation that includes each attribute
- For one-to-many relationships a copy of the primary key attributes is passed from the parent to the child relation as a foreign key

[27]

In addition to this, some of the attributes associated with the 'Support call' entity may themselves need to be promoted to entities. The 'Action' attribute is multi-valued and requires promotion to an entity with the 'Call_ref' primary key attribute of the 'Support Call' entity passed as a foreign key attribute. The 'Problem_category', 'Status' and 'Priority' attributes must contain values held within an attribute domain which requires some form of integrity constraint. There are a number of ways that this can be accomplished. Firstly, within the DBMS, in this case it will be Microsoft SQL Server, check constraints can be utilised to ensure that the values entered are from the required domain.

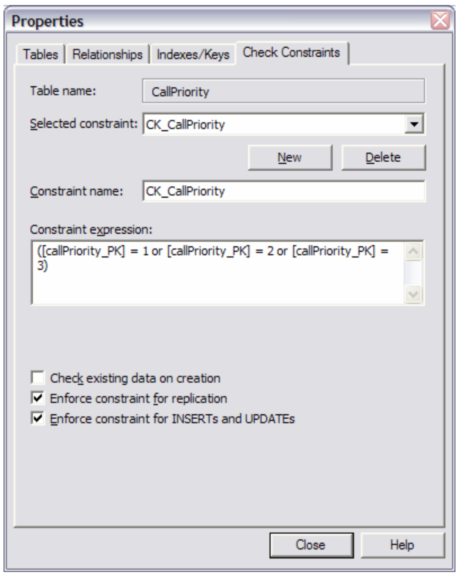


Table name: CallPriority
Selected constraint: CK_CallPriority
Constraint name: CK_CallPriority
Constraint expression:
([callPriority_PK] = 1 or [callPriority_PK] = 2 or [callPriority_PK] = 3)

- [ ] Check existing data on creation
- [✓] Enforce constraint for replication
- [✓] Enforce constraint for INSERTs and UPDATEs

Figure 6.4: Check constraints in SQL Server

Another method is to promote the attribute to an entity in the design and pass the primary key of the parent entity into the new entity as a foreign key attribute. For the helpdesk system this method provides a significant advantage over check constraints. Because the constraints are stored in a lookup table, the onus to maintain these constraints can be removed from the developer and passed onto the user. This will create an environment in which the system can be updated dynamically instead of requiring the skills of a database programmer. However, by choosing this method, the database design is made more complex.

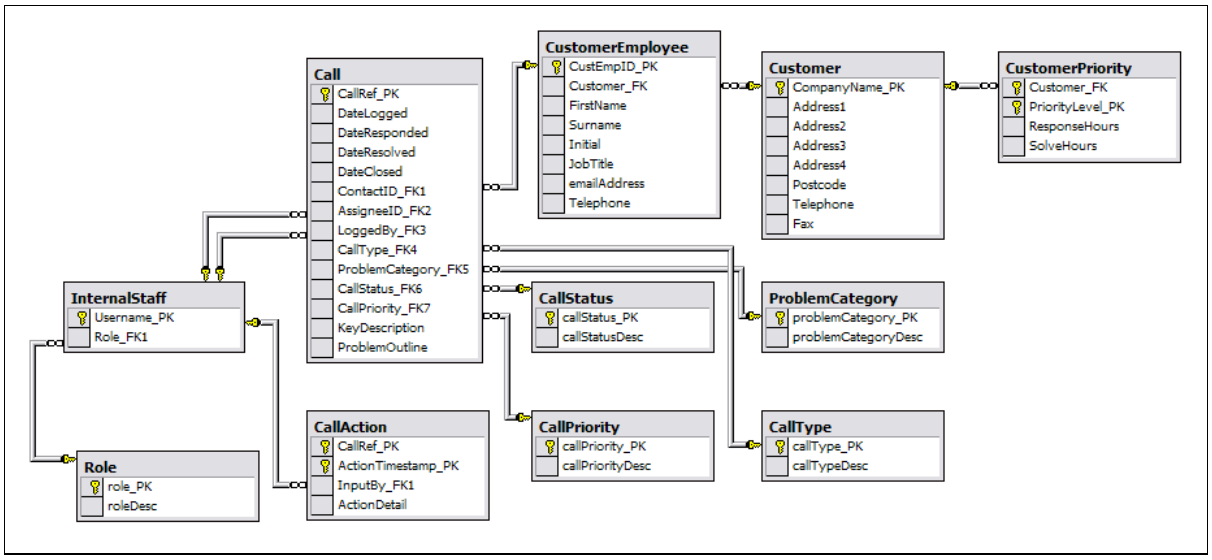


Figure 6.5: Logical Database Design

There are a number of other design considerations that need to be taken into account when creating the logical database design. The way in which the conceptual design models the users of the system over complicates the database schema. Instead of having two entities named ‘Helpdesk Administrator’ and ‘Call Assignee’, one entity named ‘Internal Staff’ is more appropriate. A member of staff then has the attributes of ‘Username’ and ‘Role’. The reason for the choice of these attributes is because the environment that the database is sitting in must be considered. Because the system is to integrate with Active Directory to deal with user authentication, it generally uses usernames as opposed to their full name. Though it would be possible to deal with full names, this would make the implementation of the system more complex.

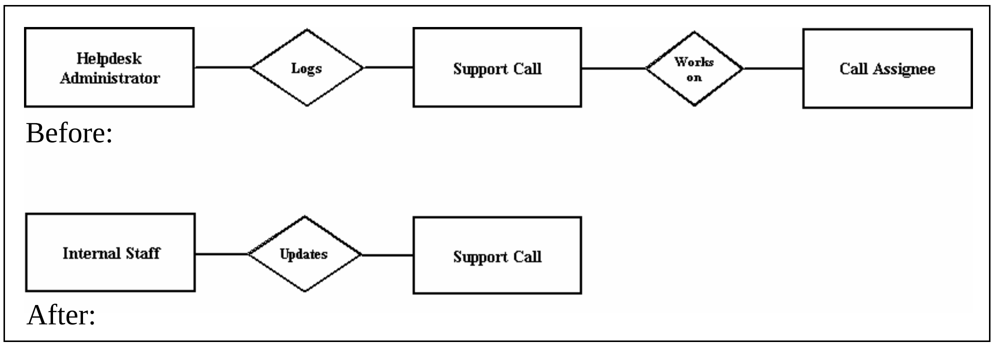


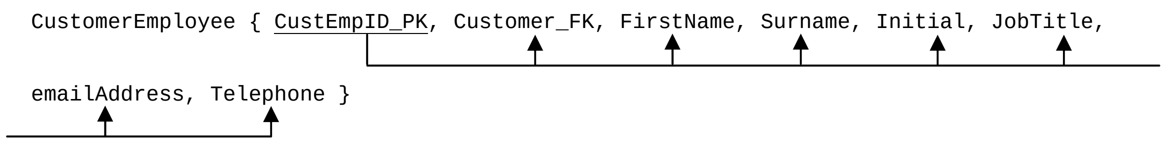
Figure 6.6: Before and after redesign

Physical Database Design

Having decided on a suitable logical design, the physical database design is reasonably easy. A full script containing the implementation of the database can be found in Appendix K; this specifies the entire database schema including the data type of each attribute.

Normalisation

By following sound database design techniques using E-R modelling, the database should be fully normalised, that is, it is in Boyce-Codd Normal Form (BCNF). However, it is possible that mistakes may have been made during the modelling process so it is therefore necessary to check each of the functional dependencies in each relation. A functional dependency “describes the relationship between attributes” [24] and when it defines a superkey of the relation, the relation is said to be in BCNF [27]. Appendix L shows the functional dependencies for all of the relations in the database. Here, the relation ‘Customer’ is used to outline the method of achieving BCNF.



```
CustomerEmployee { CustEmpID_PK, Customer_FK, FirstName, Surname, Initial, JobTitle,
emailAddress, Telephone }
```

FD1: { <u>CustEmpID_PK</u> } → Customer_FK, FirstName, Surname, Initial, JobTitle, emailAddress, Telephone

FD2: { emailAddress } → Customer_FK, FirstName, Surname, Initial, JobTitle, Telephone

FD3: { Customer_FK, FirstName, Initial, Surname } → JobTitle, emailAddress, Telephone

Here the decision to promote the use of an artificial key is justified by accounting for the consequences of the two functional dependencies FD2 and FD3. FD2 displays how ‘emailAddress’ could be used to determine the values of the other attributes. However, it falls apart when the assumption that each member of staff has a unique email address is contradicted. Likewise in FD3 it is unsafe to make the assumption that in any one organisation there will not be anybody with the same name. However short the odds, it is still a possibility that this could happen, and as such must be avoided. In short, the use of the artificial key ensures that the database schema adheres to BCNF constraints and that.

6.3 Functional Design

Modelling of the data structures has specified exactly what information is stored in the new system. This section details the processes that must be undertaken in order to store that information. It is important to get a good understanding of how processes interact with one another but because prototypes are being utilised, the application design should be considered only as a guide.

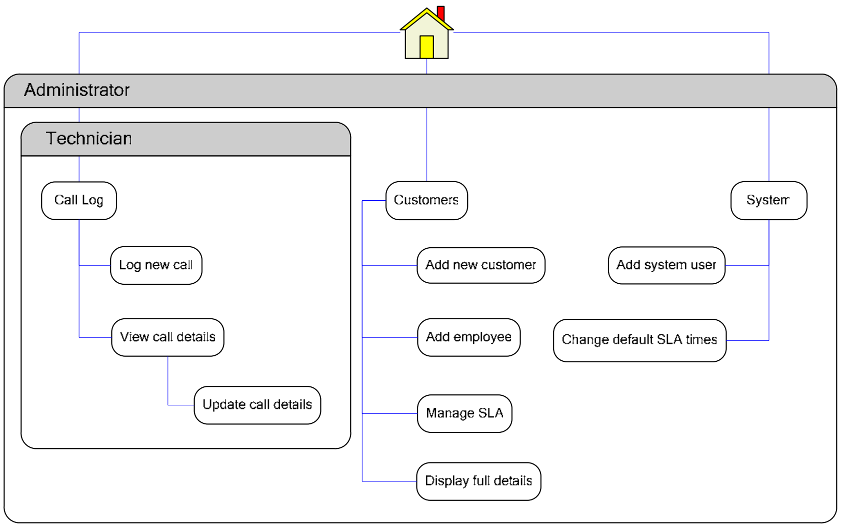


Figure 6.7: Site structure

The structure of the site is determined very much by the use case diagram formed during the requirements analysis stage (see Figure 3.2). From this use case diagram, the technician has the following requirements:

1. View support calls
2. View customer details
3. Add actions to a support call

In the design of the system, these use cases have been combined into one section of the site titled ‘Call Log’. Sub-pages off the call log will allow the technician to view the details of a call and add actions to that support call. It has been decided that requirement 2 above is only required when the technician is dealing with a support call. For this reason the design of the ‘View call details’ page must include an area that displays the details of the customer so that they can be easily contacted.

The Helpdesk Administrators and Delivery Directory inherit the functionality of the technicians but are provided with more facilities to manage the support call process. Figure 6.7 displays this extra functionality which again relates to the use cases of these actors.

The Delivery Director has an additional use case labelled ‘View call Statistics’. In the first prototype of the system, this functionality will not be implemented. However, if it was to be designed into the system, it would be an additional top level page. This means that it would be an easy addition in a future version and would not require vast overhaul of the system.

Call Log

Central to the entire system is the call logging functionality. This page displays the calls and will allow the user to filter the results depending on a set of criteria (see Figure 6.8). From the page it will be possible to log a new call and to enter any calls that have already been logged. This will then enable the users to enter further details to the call.

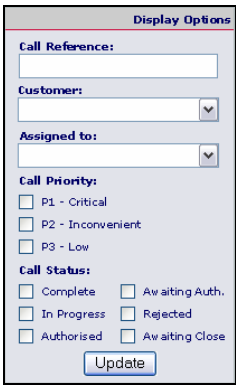


Figure 6.8: Call log filter

Customers

As was mentioned in Chapter 4 there are no systems that make it possible to log calls for a number of different clients. To overcome this, the customers screen allows administrators to add new customers and to attach employees to them. This then enables the Administrators to log new calls for these clients. In addition to this, each customer will be able to have a Service Level Agreement configured either to the system default settings or to their own personally agreed contract with Diagonal Solutions.

System Maintenance

The system maintenance screen will simply allow the core functionality of the system to be maintained and configured. The global settings such as default SLA response times, default priorities will be configured here as well as adding and removing authorisation for certain users within Diagonal Solutions' network.

6.4 User Interface

With the aim of ensuring an acceptable user interface that addresses the HCI issues discussed in Section 5.1 a page template has been developed using the popular graphics package *Macromedia Fireworks 2004.* This design has not included the use of any HTML mark-up but instead simply positions components on the page where they are considered to be most suitable. Figure 6.9 outlines the main page template and where the page details are a little more complex, further images have been created to aid the implementation of the pages.

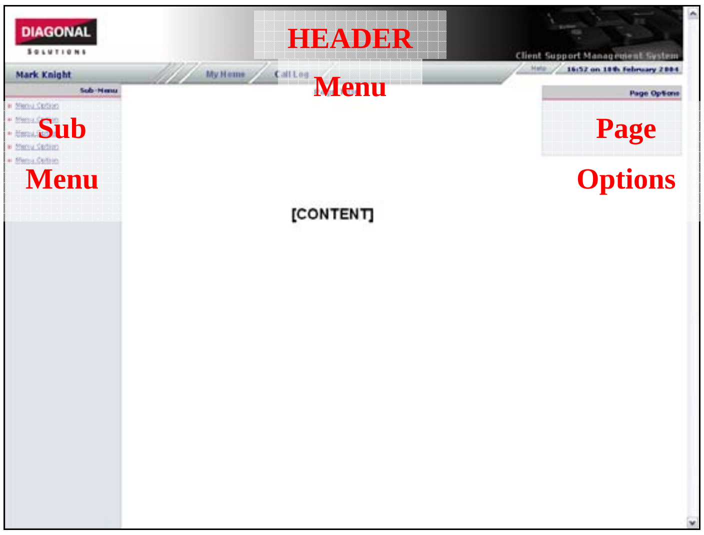


Figure 6.9: Page template created in Macromedia Fireworks

The page template is divided into five distinctive areas, the header, menu bar, sub menu, page options menu and the main content. The header and the menu bar will remain the same throughout the site and will not change. The sub-menu and page-options sections will be dependant upon the selected page and the content will clearly be the content of the page selected.

6.5 Email Reminder Engine

The *Email Reminder Engine* required in the system will send an email out to members of staff that are about to breach the service level agreement on the calls that they have assigned to them. This engine is not suited to a Web application because it would require somebody to have the page open all the time. Because it is a service that must run constantly (during business hours) a Windows Application running as a scheduled task will be developed.

Before the engine can be created rules must be defined for when the emails must be sent, and when reminders should be terminated. These are variables that may change over time so a configuration screen should be provided in the 'System' section of the site.

The process of sending reminders will be:

- Read database
- Find calls that have had no update since being logged and are X minutes from breaking SLA
- Repeat the reminder every Y minutes
- Send a final email when SLA has been broken
- Repeat every minute of the working day (defined in customers SLA)

Chapter 7

System Implementation

7.1 Development Environment

Before the implementation of the system, and in addition to the minimum requirements of the project, a development environment has been created, instead of having to use the internal infrastructure at Diagonal Solutions. There are several reasons behind this decision. Firstly, if the infrastructure at Diagonal Solutions was utilised either the development would have to take place in the office or a connection through their VPN would have to be made. This would not be satisfactory because the bandwidth of the connection restricts what can be done and it would not be practical to visit the office on a daily basis. The second reason for this decision was that in implementing the system, it is not necessarily known what the impacts on other systems might be. It is extremely dangerous to make changes on business critical systems unless full testing has been undertaken and this is a risk that the author was not prepared to take.

It is not entirely necessary for the environment at Diagonal Solutions to be replicated for the development of the first prototype. However, by undertaking this task now as opposed to future iterations of development, it is possible to build a higher fidelity prototype which encompasses more functionality such as email and the integration of Active Directory. This is beneficial for evaluation of the new system because, from as early a stage as possible, users will have an understanding of the functionality that will exist in the final product. Possibly more compelling than this is that by creating the environment that the system will sit as soon as possible ensures that the final product will work within the infrastructure at Diagonal Solutions without the need for excessive reconfiguration or coding.

Though not ideal and not identical to the infrastructure at Diagonal Solutions, the development environment consisted of a Personal Computer of reasonable specification connected to a broadband connection through a firewall and router. In order to configure Active Directory and the email server, Microsoft Exchange, a domain name was required. For the purposes of the project, the author already had an unused domain name, skiint.com, and decided to use it in this environment. In doing so it is, in essence, the equivalent to Diagonal Solutions using their domain name, diagonal-solutions.co.uk.

Once the operating system configurations are made and Active Directory appropriately configured the Exchange server is installed. When setting it up it is necessary to alter the settings of the domain name which has to be done with the domains registrar. The DNS settings on a domain translate its name (skiint.com) to an IP address (i.e 111.222.111.222). However, there are many different DNS configuration settings on a domain name. The one that needs changing is the MX (Mail Exchange) record that needs to point at the new server. Having made this configuration, it is now possible to send and receive email from the domain skiint.com. The requests for email services are all dealt with by the Exchange server which hosts mailboxes for all people on the network. The Exchange server also deals with all internally routed email, for instance, if Joe Bloggs wanted to send an email to Fred Smith and they both work within the same organisation, the email would be sent directly instead of having to go through the external Internet.

As discussed in Section 6.1 the system relies on many different server components. At Diagonal Solutions these are generally hosted on different servers, however, in this development environment, they are all packed onto the same machine. Whilst this may have adverse effects on the performance of the system, it is not a release environment and only has one person interacting with it. As such the performance issues are not apparent, although it is not a true reflection of the final use of the application where many users may interact concurrently with the system.

7.2 Data Access

The underlying requirement of this system is to store and retrieve information captured at different stages of the helpdesk procedure. In order to achieve this access to the database must be provided to the web front end of the system. Having chosen the technology for this, ADO.NET, an easy way of calling common methods is to create a data class. By including the class in the namespace of the application, it is possible to call data access functions at any time which significantly decreases the development time and code duplication. The access to the database enables SQL queries to be executed and a DataSet returned to the application. Stored procedures are used in SQL server so that query strings are not required in the code. This creates two advantages. Firstly, less data is transferred between the client and the server so the execution is more efficient. Secondly, the detail of the query is hidden from the application as so another layer of abstraction is formed. This deskills the SQL required but ensures that any developer can still work with the database [29].

7.3 Page Template

The implementation of the system requires the page design from Section 6.4 to be translated from a simple two-dimensional image into a fully rendered HTML page. All the images are sliced into relevant sizes and placed into a generic HTML page which is then used as the basic template every time a new ASP.NET (.aspx) file is required. The newly created .aspx file can then have its main content inserted within the development environment depending on which page is being implemented.

The Object Orientated approach of ASP.NET allows for inheritance of classes that lends itself to the creation of these page templates. Figure 7.1 illustrates how the page template structure works. As was previously mentioned, all standard pages in an ASP.NET application inherit methods and attributes from the `Page` (`System.Web.UI.Page`) class. Instead of the pages of this application inheriting from the `Page` class, they inherit from the `PageBase` class that includes additional cross-site functionality.

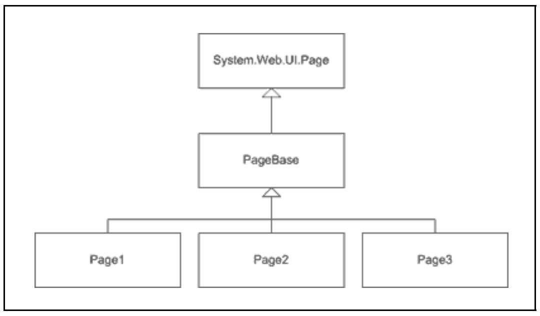
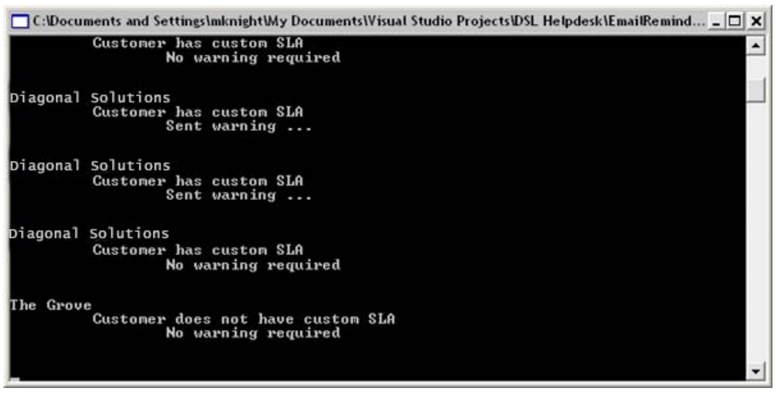


Figure 7.1: Page inheritance [28]

Web User Controls form another component of the page template model and are used to encapsulate common elements of code into a single component that can be used on many different pages [28]. In the implementation of this system, two user controls were created; a header (`header.ascx`) and a footer (`footer.ascx`). The footer control actually does nothing but close the page off, no functionality is included though it could easily be inserted should the requirement arise. The primary component of the entire template is the header control. This contains the functionality for obtaining the users' name that has logged onto the system, displaying the date and time, and providing the user with applicable menu items.

7.4 Email Reminder Engine



```
C:\Documents and Settings\mknight\My Documents\Visual Studio Projects\DSL Helpdesk\EmailRemind...
        Customer has custom SLA
                No warning required

Diagonal Solutions
        Customer has custom SLA
                Sent warning ...

Diagonal Solutions
        Customer has custom SLA
                Sent warning ...

Diagonal Solutions
        Customer has custom SLA
                No warning required

The Grove
        Customer does not have custom SLA
                No warning required
```

Figure 7.2: Email Reminder Engine

The email reminder engine is implemented separately to the Web based system because it must be run on a scheduled basis and not at the request of users it cannot rely on a Web page being open. As such, a scheduled task is created on the server that runs a console application. This checks the database for all open calls that do not have a response against them. It then checks who the call belongs to and performs a check against either their custom SLA or against the system default SLA. If the call is near to breaking the SLA, an email is sent to the support technician to remind them that they have an outstanding call that requires attention. If the call is not near to breaking the SLA, or the call has actually broken the SLA then no emails are sent until the next time the task is run (every five minutes).

Chapter 8

System Testing

Before an evaluation of the system can be carried out it is vital that system testing is performed so that the requirements can be measured against the achievements made. For Diagonal Solutions, testing represents a more vital stage that ensures the development carried out fulfils their requirements and that the system is adept enough to deal with every user’s requests. Two forms of assessment must therefore be carried out; firstly, internal validation testing to ensure data integrity and that the software runs without any adverse issues being apparent. Secondly, acceptance testing to ensure that the users are satisfied that the functional and non-functional requirements captured at the beginning of the project have been implemented appropriately.

8.1 Functional Testing

The sole aim of the functional testing carried out is to ensure that the user cannot ‘break’ the system by entering erroneous data. Creating a list of every function implemented in the system, and considering the different approaches users may take that could break the system is a good way of achieving this. Appendix P provides a comprehensive list of what is and what is not implemented in the system. Where certain functionality has not been implemented Diagonal Solutions have agreed that future developments will address this absence.

Because of the prototyping approach, coupled with the fact that a second iteration now needs to be developed, it is felt that the tests provide some very positive feedback about the system. One of the most significant outcomes of the functional testing is the uncovering of the fact that field validation needs to be implemented on each of the forms. It was felt through the implementation of the system that, because of the amount of work required to incorporate the functionality requested, time would be better spent implementing the functionality that the users would recognise to be more beneficial to them. As a result, field validation has been left to be more of a tidying up exercise after the system has passed through another iteration of development. The forms implemented successfully provide the functionality required with just a few omissions that do not form the minimum requirements (discussed in the project evaluation).

8.2 User Acceptance Testing

At the beginning of this project, after all the requirements capture and analysis had been undertaken, it was necessary for Diagonal Solutions to sign-off the requirements list and for them to have an understanding of what would be included in the first prototype. More important than the validation testing in Section 8.1 is the acceptance from the Users that it works as they expect it to. The last column in each of the tables in Appendix P shows whether or not the implementation has been formerly accepted by all the users questioned. This was determined by holding demonstrations with a number of users across each of the functional areas and allowing them to use the system for a number of test cases prepared beforehand.

8.3 Usability Testing

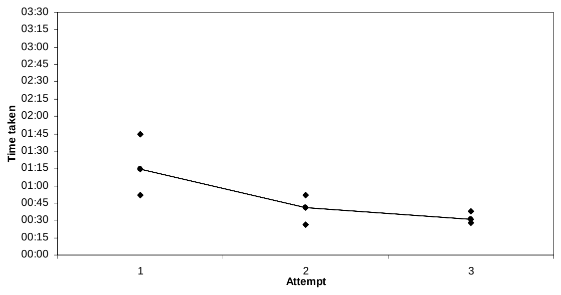
In response to the poor efforts made in the previous system to address HCI issues, much emphasis has been put on usability throughout this project. There are many defects that can be encountered with usability including the navigation, screen design, terminology, and feedback [30]. It is hoped that the careful planning and design of the user interface will pay dividends with a system that is easy to learn an intuitive to the user. However, it is acknowledged that interface design is an iterative process and that addressing the issues of being able to learn the system quickly and creating an intuitive user interface in the first iteration of development is unlikely [31]. The test plan developed uses a task based approach to determine how well the user interacts with the system; both with navigability and their comprehension of responses they receive from the system. The test does not explore every area of the system but ensures that a good cross-section of the functionality is encountered. By measuring the time it takes to complete a task, comparisons may be made to the old system. However, by running the tests more than once, it is possible to make judgements on how easy the system is to learn. Accordingly, the tests were repeated three time. The tasks performed were:

1. Change the SLA for a specified customer from the default agreement to a customised agreement
2. Add a specified employee to a specified customer
3. Log a support call for a specified customer employee and assign it to a specified technician
4. Update the status of the call logged to show that it has been cancelled by the user

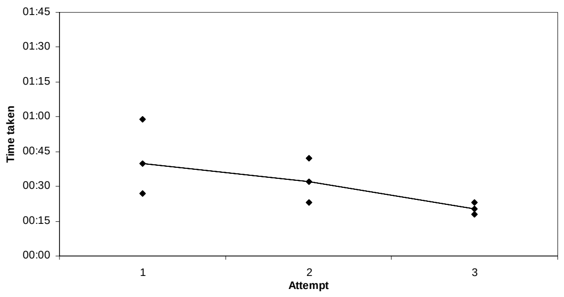
The testing was performed with a sample of four Administration employees as it is they who require the most functionality and use the system most frequently. It is also arguable that they do not have as deep an understanding with computer systems as the support technicians and as such provide a better sample for testing the learning aspects of the system.

Each of the graphs below show the results for the tasks attempted and are somewhat pleasing. The tasks were left entirely to the initiative of the staff and the graphs show a significant reduction in the time taken to complete each task (the raw data can be found in Appendix Q).

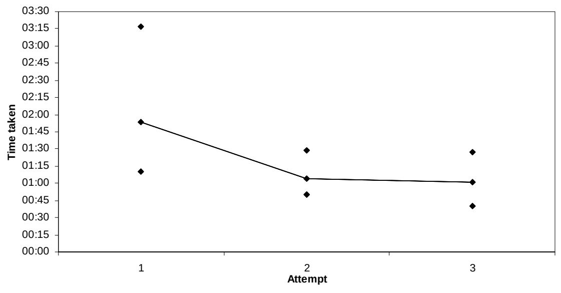
Task 1



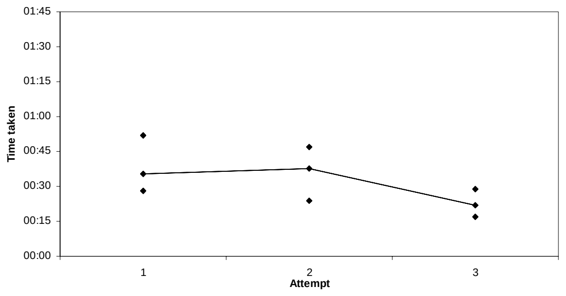
Task 2



Task 3



Task 4



Observation was felt to be an area that was not needed for the capture of the companies' requirements in Chapter 3 but is useful in the testing of the new system because the feedback is otherwise limited to users opinions. It was interesting to witness some of the mistakes made by the users when carrying out these tasks. The feedback captured from the tests provides more than the statistics above. It paves a way in which the mistakes made can be avoided in the future. The information in table … show the mistakes that were made in the usability testing and outline possible methods of overcoming these issues in future iterations of development.

Task 1

Problem	Resolution
Took a long time to navigate to the "Add custom SLA" link, though hovered the mouse over the link regularly	After speaking to the user about this link they felt that it was labelled confusingly. It is felt that the link would be better labelled as "Change to custom SLA" as, logically, nothing is actually being added it is just being altered
User went to 'System' section instead of 'Customer'	The user actually corrected the error herself and soon realised the mistake that she had made. It would not be beneficial to change the main menu headings as they were generally understood by all users. This error should be regarded as exploration of the new system as it is believed that users would soon learn what each section is designed to achieve

Task 2

Problem	Resolution
User entered the surname into the 'Initial' field and the corrected the problem.	The 'First name', 'Initial' and 'Surname' fields are displayed on above the other. By moving these in line, and reducing the size of the 'Initial' field to just one character's length, this issue should be avoided

Tasks three and four were not affected by user errors though there were some issues encountered during the tests caused by bandwidth limitations between the office and the development environment. The latency caused by this bandwidth limitation makes it impossible to make a quantitative assessment between the prototype and the system currently in place at Diagonal Solutions. However, the opinion of the staff interviewed is very much in favour of introducing the new web based system once further development has been made to address the issues discussed.

Qualitative Feedback

One of the more informative outcomes of the interviews held with the users was the qualitative feedback from both the observations and the comments made by the users. The following table presents a list of the aspects that either confused the users or were requested as enhancements and the page to which they relate.

Page	Comment
Default.aspx	Should display the call log instead of having to go to the 'Call Log'.
addCall.aspx	Call priority should default to 'Inconvenient' not 'Critical'
addCallDetails.aspx	Technicians should not be able to close the calls
displayCall.aspx	The description is not differentiated from the Action Details displayed below. It would be good if the description of the problem was highlighted so that it is easier to find the relevant information as quickly as possible.
Calls.aspx	Should be able to sort by column headings; either by date logged, customer or status of the call.

Comparison with the Old System

Given the latency issues encountered with the system testing above, a fair comparison could not be made from the office in Leeds between the old and the new systems. However, it is preferable that some comparison is made so instead of getting end users to perform the test, an 'expert' (the author) user will be used with local copies of each system hosted on the development environment. The author is considered to be an expert user because they have developed the new system and had substantial exposure to the old system. As such it is felt that the test is being conducted fairly.

Comparison between two tasks provides enough feedback to assess the usability differences between each system:

- Add the employee 'Mark Knight' to the customer 'Diagonal Solutions'
- Log a call for 'Mark Knight' at 'Diagonal Solutions' assign to 'Alex Montgomery'

On the first task, the new system performed better than the old system by ten seconds, the second task was performed just five seconds faster on the new system. This latter result is interesting. Though much of the real-estate has been cleaned up and invalid fields have been removed, a gain of only five seconds advantage is reaped. The test does however show that the user interface is improved and, from talking to people at Diagonal Solutions, is generally more pleasant to work with than the old system.

Chapter 9

Evaluation

The focus of the project must now progress from one of creation and innovation to one of contemplation. From the approach taken right through to the testing of the development work carried out, evaluating these sections provides a way in which future iterations can be embarked upon more successfully.

9.1 Project Approach

A methodology for the approach to this project was not sought after simply because one existed. It could have been decided to just use a single methodology because it had successfully been utilised elsewhere. This, however, would have been missing the point. The approach was decided on many aspects of many different methodologies that were each felt to bring individual strengths to the project. The question now is, was the approach successful? And can the project be deemed a success?

The decision to use the Waterfall model incorporating the use of prototyping was taken to try and create a structured environment that was dynamic enough to adapt to the changes around it. This method worked to a degree. The design of the system took two weeks longer than originally planned because it was deemed more necessary to build sound foundations based on an excellent understanding of the requirements than to move onto developing a system without all the required modelling having been carried out. However, taking this decision forced proceeding stages to be delayed and as such the implementation the project write-up each suffered a one week delay. So was the approach right? It is easy to say yes because it was delivered on time. It is felt that the approach was probably too dynamic with deadlines not enforced strictly enough. For a relatively complex project with limited time boundaries and numerous other factors affecting ones ability to be able to complete work on it controls needed to be applied as soon as slippage was sensed.

9.2 Requirements Analysis

As previously mentioned, getting a sound understanding of the user's requirements was reckoned to be the most important aspect of this project. It is felt that the reason for this is two-fold. Firstly, by carrying out exhaustive requirements analysis from an early stage, ideas can be brought to fruition early on in the project and as such included in prototypes that are very high fidelity. This means that what the users see in the first prototype is a good representation of what they will be delivered in the final product. Secondly, and linked to the first reason, by capturing as many requirements as possible the system can be based upon sound design and is unlikely to require significant changes as more requirements are uncovered later on.

The capture and analysis of the requirements at Diagonal Solutions was carried out early on and covered many aspects of what must be included in the product. Having known the company for approximately 18 months by this stage it is felt that a good understanding of the business and of their requirements was gained. Obtaining the requirements early on meant that the design could be carried out properly and with enough relevant information.

9.3 Technical Analysis

The technical analysis covered an extremely broad range of study from the design considerations of the user interface through to the architecture of the environment the system fits in. The analysis carried out in these areas was justly thorough and aided significantly in the design of the new system.

9.4 System Design and Implementation

The importance of a sound system design has been emphasized throughout this chapter and, having concentrated on it so much throughout the project, it was successfully achieved. The design provided a sound platform on which to build the first prototype to a high level of fidelity, that is, to a high specification that exceeds the projects minimum requirements. Choosing to implement the system in a development environment ensured that full functionality was included without having to use the infrastructure hosted at the offices in Leeds. Though it was not entirely necessary, the time spent in creating the environment at the beginning of the implementation stage has saved somebody the task of having to integrate the solution into the existing infrastructure.

Choice of technology

Ensuring that the new system functioned on the infrastructure in place at Diagonal Solutions was the key motivator to researching the technology that was available for this project. The focus, and outcome of the analysis may have been somewhat biased towards Microsoft technology but this is easily justified by the fact that nearly all of the software Diagonal Solutions rely on is Microsoft's.

Once the technology has been bought into, it is a costly business to then go and reengineer the infrastructure. However, as mentioned when the decision was made, ASP.NET was utilised because of the merits it brought to the project and, having now gained experience with the framework, it is felt that it was indeed the right decision to make. Though the learning curve was steep the technology was clearly suited to the helpdesk system. The design choice based on the decision to use ASP.NET of integrating the system with Active Directory worked well. Users are not required to log on to the system explicitly but security and access has been provided through the use of Active Directory. Again, the marriage between ASP.NET and the Microsoft Exchange Server worked well. Email facilities were provided through the use of the `Email` class and were entirely transparent to the user.

Though other technologies could have been used it is felt that they would have not been deployed as rapidly as was achieved by using ASP.NET. They almost certainly wouldn't have integrated into the infrastructure as successfully at Diagonal Solutions and as such, it can be concluded that the right decision was made.

User Interface

It has been discussed that the user interface must ensure that standards have been adhered to and that the feedback the users get from the system is in line with their expectations. The ability to easily learn the system was also considered to by of utmost importance. By designing the interface in a graphics package before the implementation was undertaken, aspects of the learn-ability of the system were considered before vast overhaul of the interface was required. The feedback from the testing of the first prototype is very positive though it is clear that more work is required with some of the more intricate detail of the system. Some of the menu items or page options require slight rewording but, generally, the interface has been received well by Diagonal Solutions. It is also recognised that the new interface design, despite being in prototype stages, performs more efficiently than the old layout. It is reassuring that the effort put into the design has been rewarded by these benefits and that the time spent was not pointless.

9.5 Future Enhancements

When the project was originally conceived there was a sense that it may be a little over simplistic for a final year project. After a small amount of market research opinion was quickly altered as it became apparent how a new system could bring numerous business benefits to Diagonal Solutions. Having now developed the first prototype of the system, the future vision has to be discussed in order to understand the authors' excitement with the project and the frustration that more could not be achieved in the time spent so far.

The qualitative feedback in the previous chapter provides a good basis to start this section from. The issues raised must be the first improvement to be made to the system. They don't form enhancements but fundamental changes that will provide adequate functionality with a well established user interface. However, beyond this there are changes that can be truly regarded as 'enhancements'.

When scrutinising the system with some developers at Diagonal Solutions, they were generally satisfied with the method in which the system had been implemented. They particularly like the interface and the method in which the template had been developed. The more compelling critique though, came from the improvements that could be made to the systems implementation. ASP.NET includes a number of tools that had not been uncovered during the analysis of the development environment – namely the RegularExpressionValidator and RequiredFieldValidator. The RegularExpressionValidator control enables validation to be performed on a field based upon a developer-defined rule (the regular expression) [32]. Both of these controls provide an improved method of implementing the forms in the system. When validation is implemented in the solution the RegularExpressionValidator can be used. Likewise, where required fields have been implemented manually in the system, they can be replaced by the RequiredFieldValidator control.

The final point to make about improvements that should be made to the implementation methods relate to data access. The system, in its first prototype form, would not support concurrent usage because of the data access implementation. Because the Web technology used leads to a disconnected front-end, controlling data becomes much more complex when many users are to use the same data source. Some form of record locking needs to be implemented in order to ensure that when one person makes a change no one else makes a change while they are editing the data. This is a complex area of research and will not be discussed any further, though, in the final version it is imperative that some method is implemented.

There are a number of further enhancements that could be implemented in the system that would make it an exciting contender to the products marketed and discussed in Chapter 4. Taking functionality from these items of software, such as the Statistical Dashboard would ensure that the system provided Diagonal Solutions with a solution that met their requirements entirely. However, other features that have not been seen on the market could be added. Discussed in the best practice guidelines was the inclusion of configuration management. This is something that is seen to be relevant and should really be implemented at a future date. Once this is implemented the system could go a stage further than any other package on the market by incorporating *Remote Control Clients* such as PC Anywhere, VNC or Microsoft Remote Desktop. This would not create just a call tracking package but would result in an entire support management where support technicians could receive a call and instead of having to lookup separate documentation, simply click a button and be connected to the clients system. Finally, it is felt that, although email is a powerful tool, a different method of notifying employees when they have a new call logged to them or when they are close to breaching an SLA. It has been considered that a Java application could be used to notify the users. Another option could be integrating the system with Instant Messaging technology to send messages over the Internet or even via SMS. The possibilities are exciting and endless. The prototype is a distance away from the vision, but is a good start to a product that was not available on the open software market.

9.6 Conclusion

Before it is able to fully control the support process at Diagonal Solutions, the implementation of the new system requires further development that is planned for the author when he commences employment at the company in the summer. It is hoped that at least some of the enhancements discussed can find their way to fruition and enhance not only the users experience of the system, but also improve the companies relationship with the numerous clients that interact indirectly with the system on a daily basis by ensuring a better level of service is achieved.

References

[1] HUGHES, B. AND COTTERREL, M. 2002. *Software Project Management*. Third Edition, McGraw Hill. ISBN 0-07-709834-X

[2] O'CONNELL, F. 2001. *How to run successful project III – The silver bullet*. Addison-Wesley. ISBN 0-201-74806-1

[3] CCTA, 1998. *Managing Successful Projects with PRINCE 2*. Revised Edition, The Stationary Office. ISBN 0-11-330685-7

[4] GANTTHEAD.COM, 2004. *Process/Project RAD – RAD – Rapid Application Development Process*. http://www.gantthead.com/process/processMain.cfm?ID=2-19516-2.

[5] LAUDON, K. C., AND LAUDON, J. P. 2002. *Management Information Systems – Managing the digital firm*. 322-323. Seventh Edition. Prentice-Hall. ISBN 0-13-061960-4.

[6] BENNETT, S., McROBB, S., FARMER, R. 1999. *Object-Orientated Systems Analysis and Design using UML*. McGraw-Hill. Glasgow. ISBN 0-07-709497-2.

[7] BENNET, S., SKELTON, J., AND LUNN, K. 2001. *Schaum's Outline of UML*. McGraw-Hill. Italy. ISBN 0-07-709673-8.

[8] OGC. 2002. *Service Support (CCTA) (IT Infrastructure Library)*. Stationary Office. ISBN 0-11-330015-8

[9] THE MOZILLA ORGANISATION. 2005. *Featured Mozilla Products.* http://www.mozilla.org/products

[10] DIX, A., FINLAY, J. AND ABOWD, G.D. 1998. *Human-Computer Interaction*, Second Edition. Prentice-Hall Europe. ISBN 0-13-239864-8.

[11] SHNEIDERMAN, B. AND PLAISANT, C. 2005. *Designing the user interface – Strategies for effective Human-Computer Interaction*, Fourth Edition. Addison-Wesley. Ch7. ISBN 0-321-26978-0.

[12] MERCHANT, S. *Norman's Philosophy of Design for Everyday Interaction*. http://www.cc.gatech.edu/classes/cs6751_97_fall/projects/ms-squared/ubicomp/ sam_essay.html.

[13] BEYNON-DAVIES, P. 1995. *Information systems 'failure': the case of the London Ambulance Service's Computer Aided Despatch project*. European Journal of Information Systems, 4, 171-184.

[14] WHYTE, W. S. 2001. *Enabling eBusiness*. John Wiley & Sons Ltd. ISBN 0-471-89941-0.

[15] MICROSOFT CORPORATION. *Active Directory architecture*. http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/deploy /projplan/adarch.mspx

[16] MAFFEIS, S. *Client-Server Computing*. http://media.wiley.com/assets/152/06/computer.pdf.

[17] MELONI, J. C. 2002. *PHP fast & easy web development*. Second Edition. Premier Press. ISBN 2-931841-87-X

[18] NETCRAFT.COM. 2005. *April 2005 Web Server Survey*. http://news.netcraft.com/archives/2005/04/01/april_2005_web_server_survey.html.

[19] BROWN, C. E. *ASP vs. PHP – Which one is right for you*. http://www.pointafter.com/Archives/nl0203.htm

[20] YUEN, P. K. AND Lau, V. 2003. *Practical Web Technologies*. Addison-Wesley. ISBN 0-201-75076-7

[21] GRIMER, T. 2004. *Students' Essential Guide to .NET*. Butterworth-Heinemann. ISBN 0-7506-6131-3.

[22] ESPOSITO, D. 2003. *The ASP.NET Page Object Model. One Day in the Life of an ASP.NET Web Page*. http://msdn.microsoft.com/asp.net/articles/architecture/default.aspx?pull=/library/en-us/dnaspp/html/aspnet-pageobjectmodel.asp

[23] DEITEL, H. M., DEITEL, P. J. AND NIETO, T. R. 2002. *Internet & World Wide Web How to Program*. Second Edition. Prentice-Hall. New Jersey. ISBN 0-13-030897-8

[24] CONNOLLY, T. AND BEGG, C. 2005. *Database Systems. A Practical Approach to Design, Implementation, and Management*. Fourth Edition. Addison-Wesley. ISBN 0-321-21025-5

[25] WEBOPEDIA.COM. 2005. *N-Tier Application Architecture*. http://www.webopedia.com/quick_ref/app.arch.asp

[26] TEOREY, T. J. 1999. *Database Modelling & Design*. Third Edition. Morgan Kaufmann Publishers. United States of America. ISBN 1-55860-500-2

[27] ELMASRI, R. AND NAVATHE, S. 2000. *Fundamentals of Database Systems*. Third Edition. Chapter 9. Addison-Wesley. United States of America. ISBN 0-201-54263-3

[28] PROVOST, P. 2002. *ASP.NET Page Templates – Using Inheritance*. http://www.codeproject.com/aspnet/page_templates.asp

[29] MICROSOFT CORPORATION. 2003. *.NET Data Access Architecture Guide. Patterns and Practices*. http://www.microsoft.com/downloads/details.aspx?FamilyId=0D95803A-59D7-46E2-8DFA-01905846AC67&displaylang=en

[30] LINDGAARD, G. 1994. *Usability Testing and System Evaluation A guide for designing useful computer systems*. Chapman & Hall. ISBN 0-412-46100-5

[31] CATO, J. 2001. *User-Centered Web Design*. Addison-Wesley. Italy. ISBN 0-201-39860-5

[32] SMITH, S. A. 2004. *Regular Expressions in ASP.NET.* Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/regexnet.asp

[33] MICROSOFT CORPORATION. 2005. *.NET Framework General Reference RequireFieldValidator Control*. http://msdn.microsoft.com/ library/default.asp?url=/library/en-us/cpgenref/html/cpconrequiredfieldvalidatorcontrol.asp

Appendix A – Reflection

If there is one thing I will take away with me after I hand this report in it is how important time management is. Not just for the project but for the final year in general. I was faced with the daunting task of having to juggle 50 credits of modules in the first semester with the need to do work on the project. It is vitally important that when there are small gaps in module commitments that progress is made on the project. The first thing that I would recommend to students about to undertake any project would be to utilise your spare time as much as possible. Even if all you do is spend 15 minutes on reading, it is 15 minutes you won't have to spend closer to the deadline date.

I was in a fortunate position of being able to return to the company that I worked for last year during my industrial placement. This provided me with an opportunity to not only show them what I was capable of doing for future employment but also to base my project on a real world situation. If the reader is currently deciding on a project my advice would be to pick something that you are interested in. This forms a significant portion of your degree and begins to take over your life. Don't let it be on something that you're not going to be excited by.

Having carried out the project for a third party I have seen the company I worked for very differently. Going in and arranging meetings made me feel more as a third party consultant than an employee. It was nice to be able to go back and continue the professional relationships that I had formed in the previous twelve month but it also provided me with an ideal opportunity of building skills required throughout my professional life. Advice to future students would be, regardless of whether they have worked for the third party before or not, to not be afraid of asking questions – they only aid your project and the relationship that you form with the company. I now find myself in the extremely fortunate position of having found employment upon graduation with a good starting salary.

Throughout the final year one often finds oneself reflecting on the first years of University – some may say wishing those days were still here! The project is the one true chance of be able to study something that is a genuine choice, not something that the University have enforced on you. It has given me the opportunity to explore new technologies and to research the areas that I am interested in. Don't be afraid to try new tools and to explore your own ideas.

Finally, to reiterate the point I made at the beginning, time management has been an area where I find my biggest weakness. The project has been an enjoyable yet testing experience. Don't underestimate the amount of time it takes to carry out each deliverable of your project, plan your work carefully, do as much work as you can as early as possible and you will hopefully find that you wont be reflecting on time management skills like me!

Appendix B – Interviews

Meeting with Mike Lord – 18/11/2004

The purpose of this meeting is to ascertain the overall objective of the new system. It is very open ended and is designed to answer one overriding question – what is required from the new system?

The outcome of the meeting follows.

The new system must:

- Have the ability to log a call – with type and description fields
- Be able to track a call
- Be able to escalate the call including deadlines
- Report against clients calls
    - Success of handling calls against SLA
    - Track against SLA
- Log how a call was resolved

It would be nice to have a real-time graph showing different statistics.

The system could eventually be used on a number of different devices (mobile/pda/tablet)

Meeting with Marc Teale – 14/12/2004

The purpose of this meeting is to ascertain the scopes of two parts of the proposed system. Firstly, we must determine exactly how the system will fit into the current IT infrastructure. Secondly, it is important to capture the requirements of a member of the infrastructure team. Being one of the main users of the system, it is necessary to determine what it is the current system doesn't do, and what other additions they require.

Diagonal Solutions Infrastructure

The proposed system will be based on web technologies and as such, an Internal web server (Intranet) will be required (IIS).

Will there be any issues faced by integrating the new system with the current Active Directory?

> *As long as no changes are required to the schema or to the entries, this would not propose any problems. If you are asking if A.D should be used for security this would not be a problem but the user should not be prompted for a password unless they are not in the office*

The system may require users to be a member of some group (e.g 'HelpdeskUsers'). It could also require administrators of the software to be in a separate group ('HelpdeskAdmin'). Would this be an issue?

> *No*

What version of Exchange is Diagonal Running?

> *Exchange 2003 runs on Wormtongue*

What version of Active Directory is Diagonal Running?

> *Windows 2000 Server on Radagast and Pippin*

Which server will be used to host the web pages?

> *Eowyn, it also hosts CRM and SharePoint portal server*
>
> MK: Could this cause problems?
>
> > *No, there should be no problems adding any other web applications to the server*

What version of Windows / IIS is that server running?

> *The version packed with Windows Server 2003, Vs6*

Which server will be used to host the databases?

Aragorn,

What version of Windows and SQL is that server running?

It runs SQL 2000 on Windows Server 2003

Infrastructure Requirements

The new system will need to include everything that the system currently does with the addition of the following.

Emails

No additional functionality required

Reminders

Once the calls is logged if no action has been taken on the call it should send email again and say no one has worked on it. The length of tie it waits would be determined somewhat by the SLA of the customer

Search facility

We should be able to search on the problem and description fields of the call.

Meeting with Julie Jowett and Paul Buckley (Admin)

The purpose of this meeting is to ascertain the problems encountered with the current system and how they may be resolved. It has also been set up to determine what functionality may be required in the new system that differs from the current way of doing things.

Should be able to view all outstanding calls and add details to them

Every week Julie email Mike a list of outstanding calls who then emails the assignees to either update the call or close it.

Current system
Description field not big enough

Must only allow call to be logged against client with a current contract (not the case – all past clients are displayed which is a pain)

Reference number should be displayed before details are entered, not when it is saved

When the window is closed there should be a prompt in case work has not been saved and the window was closed accidentally.

What does the modem field do?

What is call type?

There is no SLA failsafe

No need for comments field

Should be able to assign default values by customer

Appendix C – Use Case Description Forms

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*Create / Update customer details*
*	Primary Actor: *(Actor that initiates Use Case)*	Helpdesk Administrator
R	Other Actors:	None
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the helpdesk administrator will have stored up to the date information about the customer and be able to add employees of that customer to the system
R	Basic Course of Events: *(The Normal Flow)*	This use case begins when the helpdesk administrator is made aware of a new customer. All the relevant details are noted down by the administrator and processed into the system
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	A current customer may need their details updating on the system (e.g new telephone number).
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	None
	Assumptions:	
	Pre-conditions:	Helpdesk administrator is set up on the system
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*Create / Update customer staff details*
*	Primary Actor: *(Actor that initiates Use Case)*	Helpdesk Administrator
R	Other Actors:	None
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the Helpdesk administrator will know who works for the client and will be able to log support calls for them
R	Basic Course of Events: *(The Normal Flow)*	This use case begins when the helpdesk administrator is made aware of a new member of staff being employed by one of Diagonal Solutions' customers. Their details are noted down and input into the system
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	
	Assumptions:	
	Pre-conditions:	The Customer has been set up on the system
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*Create / Update Support call*
*	Primary Actor: *(Actor that initiates Use Case)*	Helpdesk Administrator
R	Other Actors:	
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the helpdesk administrator can manage the support call effectively by liaising with relevant technicians/
R	Basic Course of Events: *(The Normal Flow)*	A customer calls the helpdesk with a problem. The details are taken down and saved in the system. The call is then assigned to a technician to be actions
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	A customer may email a call instead. The helpdesk administrator may have been assigned the call back from a technician for the call to be closed
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	The support call requested by the customer is not in the covered by the contract they purchased.
	Assumptions:	
	Pre-conditions:	Customer and employee have been entered into the system.
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*View Support calls*
*	Primary Actor: *(Actor that initiates Use Case)*	Helpdesk Administrator
R	Other Actors:	Support technician & Delivery Director
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the actors are made aware of calls logged to them and other people in the organisation
R	Basic Course of Events: *(The Normal Flow)*	The actor requests to see calls logged to them depending on certain criteria and the calls are displayed.
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	There are no support calls
	Assumptions:	
	Pre-conditions:	
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*View Customer details*
*	Primary Actor: *(Actor that initiates Use Case)*	Helpdesk Administrator
R	Other Actors:	Support technician & Delivery Director
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the actors are made aware of how the customer can be contacted
R	Basic Course of Events: *(The Normal Flow)*	The actor requests to see the customer's contact details. They are displayed
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	Technicians only see the contact details when they need to call the customer. They only need to contact the customer when they are dealing with a call so this is the only time they may request the details
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	
	Assumptions:	
	Pre-conditions:	
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*Add actions to support call*
*	Primary Actor: *(Actor that initiates Use Case)*	Support technician
R	Other Actors:	Helpdesk Administrator & Delivery Director
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case an up-to-date picture of the customers problem is maintained. The user does not forget what they've done
R	Basic Course of Events: *(The Normal Flow)*	The support technician makes changes to the customers system. He then logs these changes against the support call
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	A Helpdesk Administrator or the Delivery Director need to add details to the call regarding non-specified circumstances (e.g Requests for more details, call not covered in contract)
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	
	Assumptions:	
	Pre-conditions:	A support call has been raised
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

*	Use Case Name: *(The name as it appears in the Use Case Model)*	*View Call Statistics*
*	Primary Actor: *(Actor that initiates Use Case)*	Delivery Director
R	Other Actors:	Helpdesk Administrator
R	Value Proposal to Actor(s) *(the goal of the Use Case from the Actor's perspective)*	Upon completing this use case the actor has a clear picture of how well the support desk is functioning
R	Basic Course of Events: *(The Normal Flow)*	The actor requests to see statistics (reports) about the system. They specify certain criteria and are produced a report.
	Alternative Paths: *(Other paths through the use case which result in a successful outcome – typically variations to the basic course of events, determined by the actor and their needs).*	
	Exception Paths: *(Other paths through the use case which result in an unsuccessful outcome – typically when something goes wrong)*	
	Assumptions:	
	Pre-conditions:	
	Post-conditions:	
	Related Business Rules: *(Reference to your Business Rules list)*	
	Related Non-Functional requirements – Usability, Performance, Security: *(Any non-functional requirements that are specific to this Use Case rather than the system as a whole)*	
*	Project:	Helpdesk Management System
*	Author:	Mark Knight
*	Date:	

A Full Use Case Description should fill in all boxes, write None if appropriate.
** = Mandatory whenever this form is used, R = Recommended whenever this form is used*

Appendix D – Service Level Agreement document

Service Definition	
Service Name	**SLA001 – Help Desk**
Service Description	This service provides the Coal Authority with access to, and use of the Diagonal Solutions' Help Desk.
Service Benefits	Single point of contact for all IT issues. Time logged against calls can be monitored. Calls can be checked against previous occurrences. Ensures resources are assigned according to business impact. (See notes).
Process Summary	

Service Parameters	
Availability	08:00 to 18:00 Monday to Friday except public holidays. No service outside these hours.
Response Time	Calls will be acknowledged within 1 working hours, either verbally or via Email. Diagonal Solutions will aim to resolve all support calls thus: Critical issues within 4 hours with engineer/analyst work through. For critical calls that cannot be resolved within 4 hours using normal procedures all reasonable endeavours will be made by Diagonal Solutions to find a resolution to the call within the next working day. Moderate issues within 2 working days Low priority issues within 5 working days.
Turn Around	Diagonal Solutions will assign appropriate technical resource to the resolution of the issue, taking the following factors into consideration: ▪ Customer Impact (Severity) ▪ Supported Status (Schedule Classification) ▪ Previous Occurrences (History)
Scope	The schedules of Directly-Supported, Indirectly-Supported and Unsupported software will provide a definitive list of supported software. See Appendix A

Volume	There is a limit of 3 calls per week (up to 156 per annum) that will normally be accepted under this agreement. However, Diagonal Solutions will periodically review the time spent by employees in the resolution of support calls and reserve the right to modify future premiums accordingly.
Access	Any designated Coal Authority employee (up to a maximum of 3) and the Cap Gemini Helpdesk are entitled to call the Help Desk
Notes	Each call will be assigned one of three priorities: Appendix A Critical = System down or other serious business-impacting issue. This equates to Coal Authority calls of priorities 1 and 2. Appendix B Moderate = Users affected but business not seriously impacted. This equates to Coal Authority calls of priority 3. Appendix C Low = A problem that does not prevent user operation.

Service Accountability	
Reporting	As part of the monthly report, all service parameters will be reported and any variances from the agreed service levels noted.
Contact	Tel: (0113) 220 8379 or Email: support@Diagonal-Solutions.co.uk
Responsibility	The primary IT Contact onsite must inform the Diagonal Account Manager of any predicted change to business plans or technical implementation which may affect the way this service is delivered.
Review Period	No formal review will be carried out. All service issues should be raised with the Diagonal Account Manager on 0113 220 8377.
Predicted Changes	There are currently no major developments, which are predicted to significantly impact this service.

Appendix E – Internal Documentation

Support options and strategies

Problem Logging

As soon as a support call comes in, it receives the attention from the help desk staff that classifies the problem according to the impact it is having on the customer. The problem is then emailed to the most appropriate person. The help desk staff have a skills matrix in place which assists them in assigning the support call to the most appropriate member of staff. Where the priority of the call is anything other than "low" a verbal call is made to the staff member to verify that they have received it. The support call is logged on the incident recording system which records, amongst other things, the time the call is received, it's priority, the customer contract name and phone number, to who within DS the call is assigned, and when a follow-up call to that engineer should be made. This last field enables a "double-check" to be made which helps to ensure that all calls are managed. The responsibility for the resolution of the call rests with the engineer to whom the problem is assigned.

Problem appraisal

The Engineer will then immediately read the text of the support call and appraise the problem. The SLA outlined later in this document is followed and at this time the call status may be elevated or reduced as necessary. This is the primary problem appraisal. The engineer will then decide what action to take or whether to pass the call onto a third party. Responsibility still lies with the engineer and the support call is updated to reflect any and all action that is taken. The nature of the problem (Hardware, Network, Operating System, Application etc) may be determined at this point.

Problem confirmation

Within the limits set out in the SLA, when the engineer comes to investigate the problem, a first call will be made to the Council to confirm the nature, scope and detail of the problem. This forms the second appraisal of the issue and status may be elevated or reduced as necessary. The support call log is updated with any action taken. If not already identified, the nature of the problem (Hardware, Network, Operating System, Application etc) is normally determined at this point.

Problem Diagnosis

The engineer actually dealing with the problem will allocate time to the resolve the issue, within the limits set in the SLA. The following approaches are used to assist in the resolving issues:

Diagnostic Tests

It is often the case that tests need to be carried out before the source of the problem can be diagnosed. If this is the case the engineer will either talk the customer through performing those test (where telephone-only support exists) or perform those tests himself (this covers the majority of customers). In order for the engineer to carry out his tests the following options are available:

Support Modem – The customer has a dedicated or shared modem to which DS can dial and take control of the server or workstation in question. Such access is protected by passwords and preliminary verbal confirmation.

Routed Solution – The larger customer enjoy a routed solution to DS. Any PC on DS' network (when properly configured) can connect to the customer servers and/or workstation using networked ISDN routers. The routers are protected by passwords for both access and dial-up

Standalone PC – Where it is not practical or permissible to interconnect customer networks and DS' a number of standalone PCs with modems are used solely for dial-up support purposes. This satisfies the security requirements of even the most stringent of DS customers. This scenario covers most public-sector customers.

Professional Knowledge

DS employ highly competent engineers with a good deal of professional knowledge. In many cases the engineer will know about the issue simply due to his experience.

Previously Encountered Issues

DS maintain a searchable technical customer database that details the configuration of each customer site. This database contains documents, diagrams, and links to other resources. If the problem has previously occurred at another customer site, this search will reveal this fact and help to provide a speedy resolution.

Problem sharing

DS' engineers enjoy a professional working environment where information exchange and the proliferation of ideas are encouraged. If the engineer in question does not know the answer, it is highly probably that one of his colleagues will.

Internet Support

DS engineers are equipped with knowledge not only the facts regarding support issues but with knowledge of how to search for those facts. Simply knowing the location of a support site is not enough. Engineers know hot to interrogate support sites in order to gain the answers that are required. This is another area DS vigorously pursues in recruitment policies.

Third Party support

DS are a Microsoft Certified Partner and many engineers are Microsoft Certified Professionals. DS subscribes to the MSDN programme and has a support relationship with Microsoft, which allows access to knowledge not publicly available. DS have support contracts with numerous other third-party software suppliers

<u>Issue Management</u>

All support calls are the overall responsibility of the DS Delivery Director and will be escalated to that level as necessary under the SLA. Once again, the support call log is updated at all level, whenever there is work done on that particular support call. This includes contracts with third parties.

Impact Status

Since support calls have different impacts to deifferent businesses, and due to the sheer variety of topics that support calls may cover, Engineers will allocate time accordingly. For this reason, DS only operate 3 levels of severity:

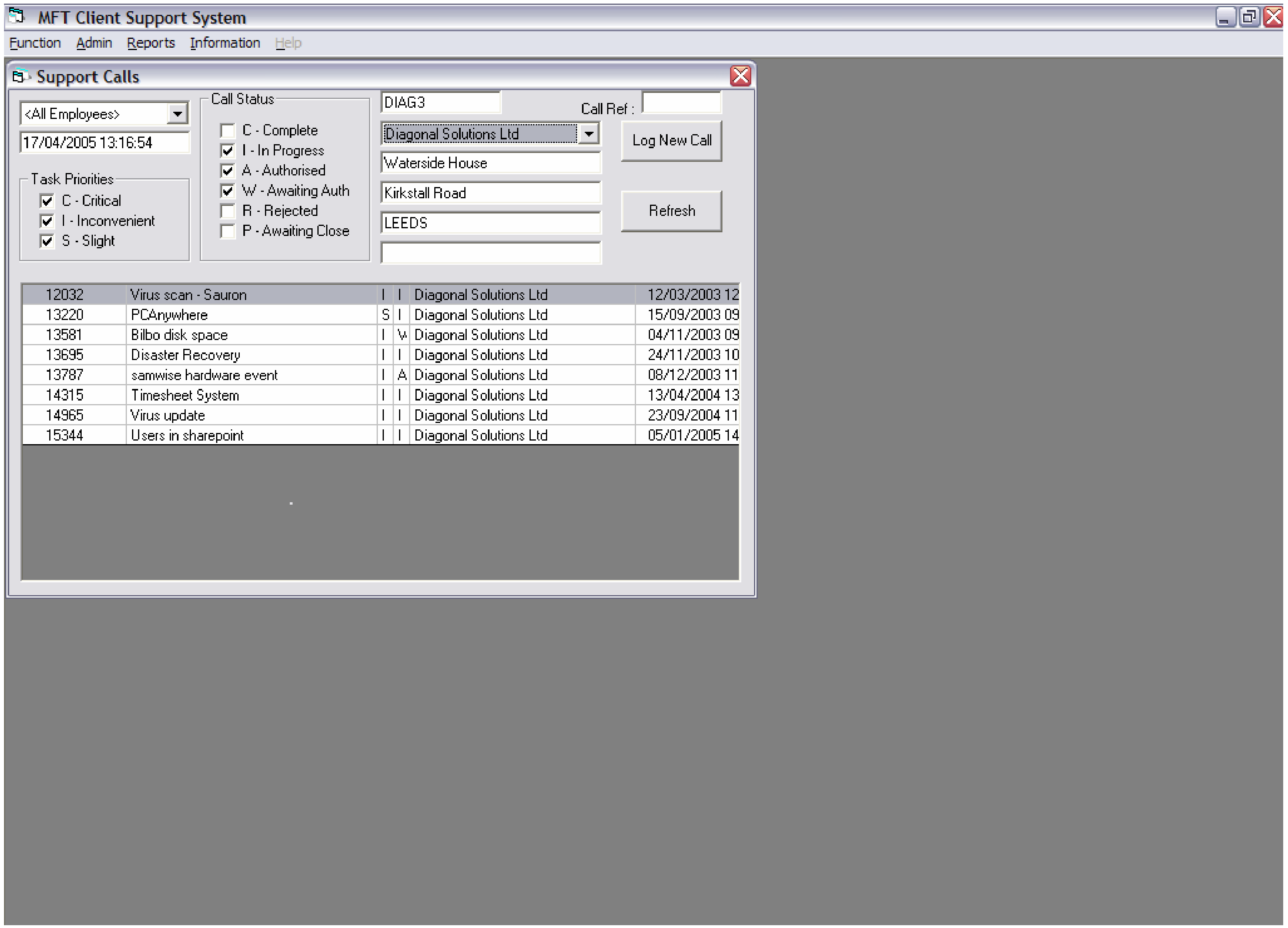
Severity	Response Time	Description
Critical	4 hours	The issue is preventing the customer from doing business, either totally or significantly. This covers things like: Mission-Critical system down, total loss of functionality with a field etc.
Moderate	8 hours	The issue is not significantly preventing the customer from doing business, but has the potential to do so if the issue is not resolved in a timely fashion. This would normally cover issues like: Non-Mission-Critical system down, Mission-Critical system performance or functionality issues, partial loss of functionality within a field, or total loss but where a workaround exists.
Low	24 hours	The customer is not prevented from carrying on business in any way, or is only marginally inconvenienced by the issue. Mission-Critical systems are unaffected but Non-Mission-Critical systems may be experiencing performance or functionality issues. Requests for change or enhancements often start as a support call with this status.

Whilst the above response times are appropriate for most customers, some customers require enhanced levels of service. Where enhanced levels of service have been agreed, Engineers are made aware of this fact.

Escalation Procedures

Like most I.T. companies, DS cannot guarantee that a fix will be found for any given problem. DS will however pursue all reasonable endeavours to resolve any customer issues where a support contract exists. DS will guarantee that a response to a customer will occur within agreed SLA standards. DS have a demonstrable track record in resolving customer issues.

Appendix F – Current system screenshot



<All Employees>
17/04/2005 13:16:54

Task Priorities
- [✓] C - Critical
- [✓] I - Inconvenient
- [✓] S - Slight

Call Status
- [ ] C - Complete
- [✓] I - In Progress
- [✓] A - Authorised
- [✓] W - Awaiting Auth
- [ ] R - Rejected
- [ ] P - Awaiting Close

DIAG3
Diagonal Solutions Ltd
Waterside House
Kirkstall Road
LEEDS

12032	Virus scan - Sauron	I	I	Diagonal Solutions Ltd	12/03/2003 12
13220	PCAnywhere	S	I	Diagonal Solutions Ltd	15/09/2003 09
13581	Bilbo disk space	I	W	Diagonal Solutions Ltd	04/11/2003 09
13695	Disaster Recovery	I	I	Diagonal Solutions Ltd	24/11/2003 10
13787	samwise hardware event	I	A	Diagonal Solutions Ltd	08/12/2003 11
14315	Timesheet System	I	I	Diagonal Solutions Ltd	13/04/2004 13
14965	Virus update	I	I	Diagonal Solutions Ltd	23/09/2004 11
15344	Users in sharepoint	I	I	Diagonal Solutions Ltd	05/01/2005 14

Appendix G – Diagonal Solutions network topology

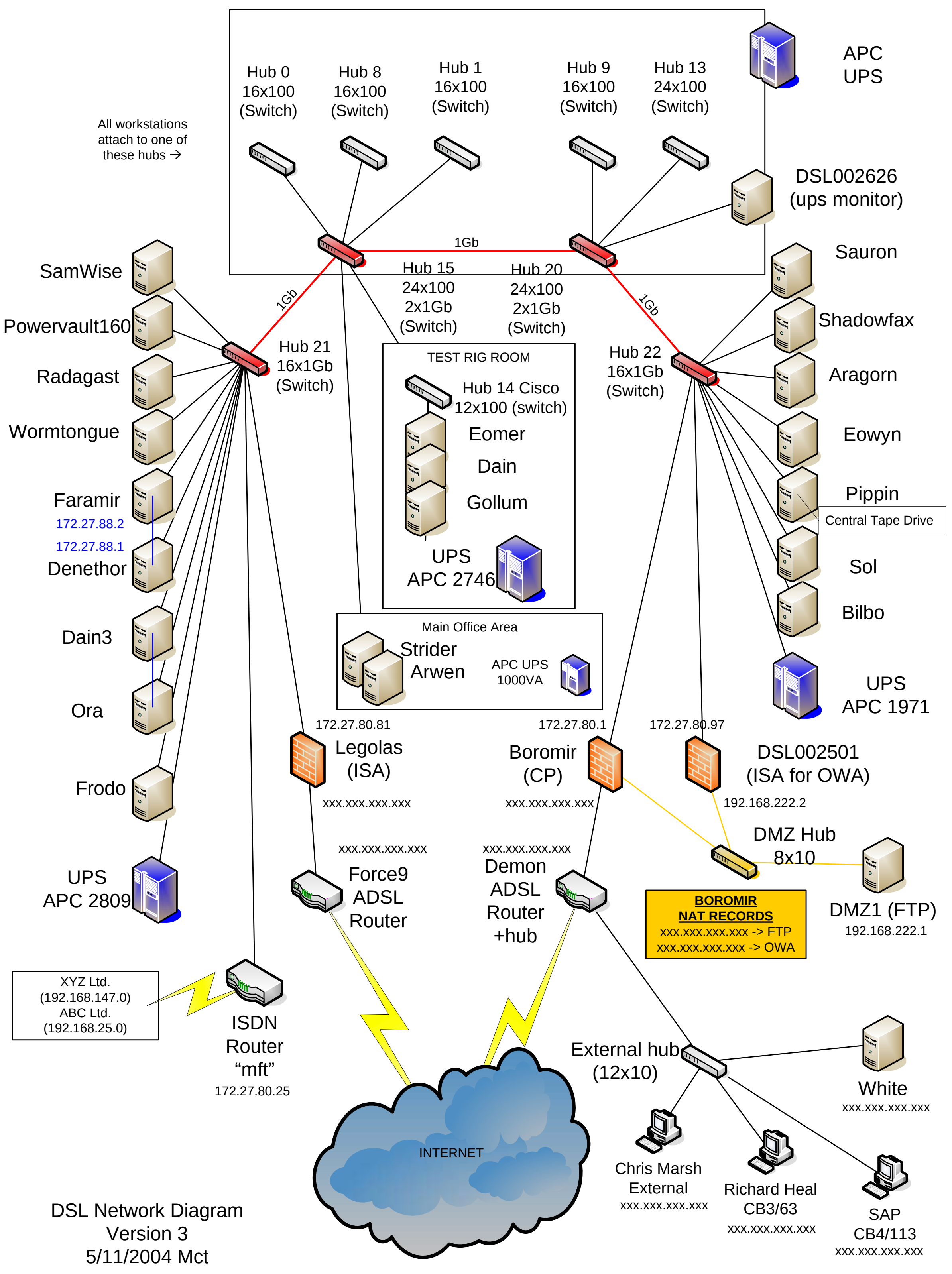


Figure 0.1: Diagonal solutions network topology (internal documentation)

Appendix H – Diagonal Solutions standard workstation build

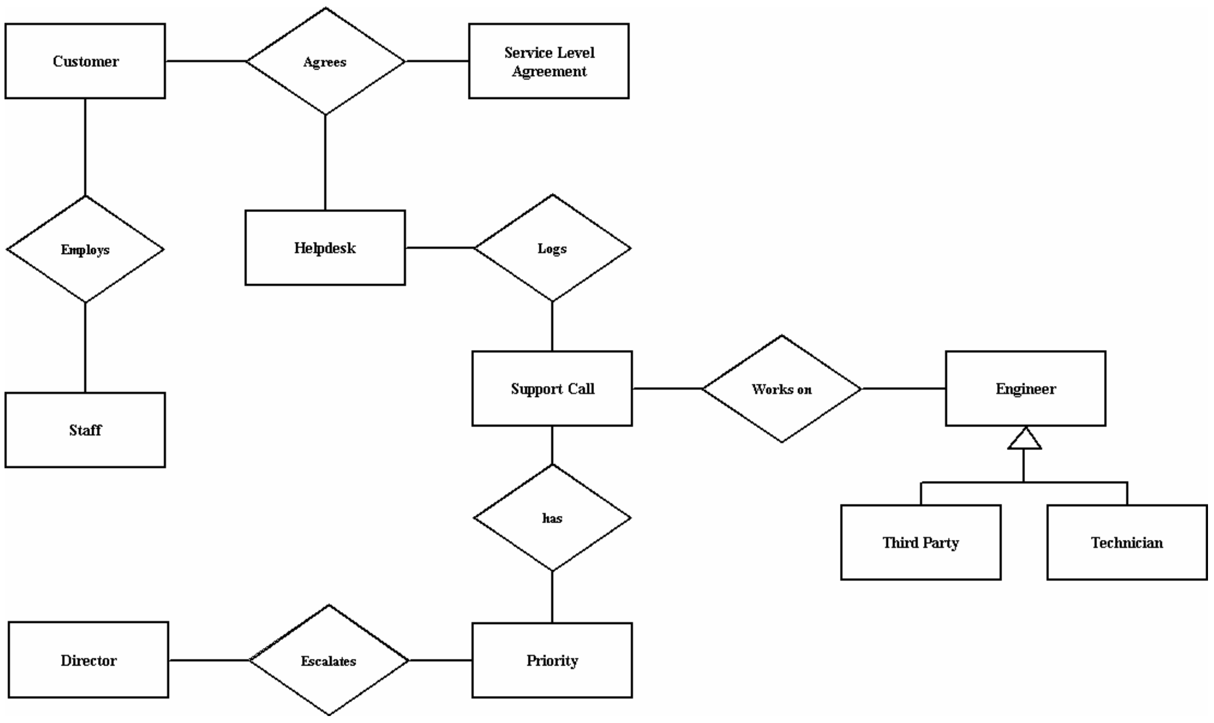
Operating System:

- Microsoft Windows XP Professional

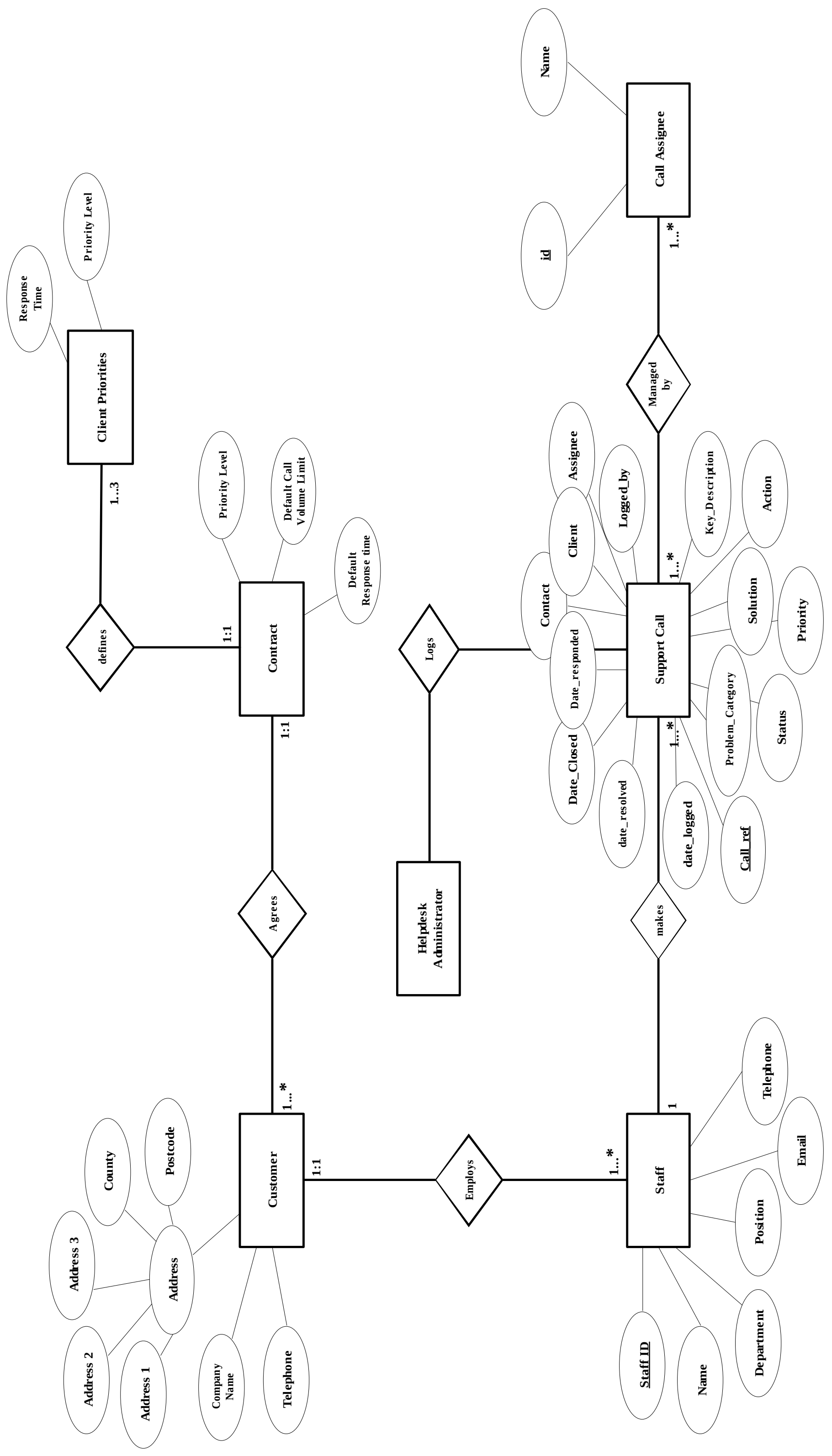
Other components:

- DirectX 9.0b
- Windows Media Player 9 series
- MDAC 2.8
- MSXML4.0 SP2
- Microsoft .NET Framework 1.1
- Windows Messenger 5.0 and point at Live Communications Server
- Office 2003 Professional
    - Word
    - Excel
    - Outlook
    - PowerPoint
    - Publisher
    - OneNote
    - Visio
    - Access
    - FrontPage
    - InfoPath
- Acrobat Reader
- Winzip
- Network Associates VScan Enterprise 7.1

Appendix I – First draft E-R diagram



Appendix J – Final E-R diagram



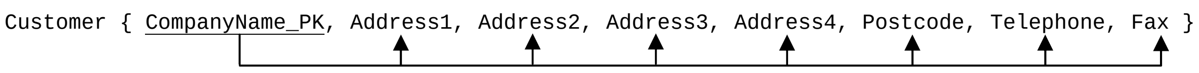
Appendix K – Database implementation script

```
/* Reference Tables */
CREATE TABLE Role (
        role_PK char(1) PRIMARY KEY,
        roleDesc varchar(25) NOT NULL
)

CREATE TABLE CallType (
        callType_PK char(1) PRIMARY KEY,
        callTypeDesc varchar(25) NOT NULL
)

CREATE TABLE ProblemCategory (
        problemCategory_PK char(1) PRIMARY KEY,
        problemCategoryDesc varchar(25) NOT NULL
)

CREATE TABLE CallStatus (
        callStatus_PK char(1) PRIMARY KEY,
        callStatusDesc varchar(25) NOT NULL
)

CREATE TABLE CallPriority (
        callPriority_PK tinyint PRIMARY KEY,
        callPriorityDesc varchar(25) NOT NULL
)

/* System Tables */
CREATE TABLE Customer (
        CompanyName_PK varchar(60) NOT NULL PRIMARY KEY,
        Address1 varchar(30) NOT NULL,
        Address2 varchar(30),
        Address3 varchar(30),
        Address4 varchar(50),
        Postcode varchar(10) NOT NULL,
        Telephone varchar(15) NOT NULL,
        Fax varchar(15)
)

CREATE TABLE CustomerEmployee (
        CustEmpID_PK int PRIMARY KEY,
        Customer_FK varchar(60) NOT NULL FOREIGN KEY REFERENCES Customer(CompanyName_PK),
        FirstName varchar(20) NOT NULL,
        Surname varchar(20) NOT NULL,
        Initial varchar(1),
        JobTitle varchar(40),
        emailAddress varchar(50),
        Telephone varchar(15),
)
```

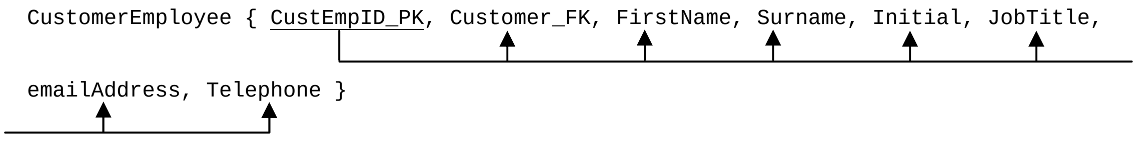
```
CREATE TABLE InternalStaff (
        Username_PK varchar(50) PRIMARY KEY,
        Role_FK1 char(1) NOT NULL FOREIGN KEY REFERENCES Role(role_PK),
)

CREATE TABLE Call (
        CallRef_PK int PRIMARY KEY,
        DateLogged datetime NOT NULL,
        DateResponded datetime,
        DateResolved datetime,
        DateClosed datetime,
        ContactID_FK1 int NOT NULL FOREIGN KEY REFERENCES CustomerEmployee(CustEmpID_PK),
        AssigneeID_FK2 varchar(50) NOT NULL FOREIGN KEY REFERENCES InternalStaff(Username_PK),
        LoggedBy_FK3 varchar(50) NOT NULL FOREIGN KEY REFERENCES InternalStaff(Username_PK),
        CallType_FK4 char(1) FOREIGN KEY REFERENCES CallType(callType_PK),
        ProblemCategory_FK5 char(1) NOT NULL FOREIGN KEY REFERENCES
ProblemCategory(problemCategory_PK),
        CallStatus_FK6 char(1) NOT NULL FOREIGN KEY REFERENCES CallStatus(callStatus_PK),
        CallPriority_FK7 tinyint NOT NULL FOREIGN KEY REFERENCES
CallPriority(callPriority_PK),
        KeyDescription varchar(100) NOT NULL,
        ProblemOutline varchar(1000)
)

CREATE TABLE CallAction (
        CallRef_PK int NOT NULL,
        ActionTimestamp_PK datetime NOT NULL,
        InputBy_FK1 varchar(50) NOT NULL FOREIGN KEY REFERENCES InternalStaff(Username_PK),
        ActionDetail varchar(2000) NOT NULL

        CONSTRAINT CallAction_PK PRIMARY KEY CLUSTERED(CallRef_PK, ActionTimestamp_PK)
)

CREATE TABLE CustomerPriority (
        Customer_FK varchar(60) NOT NULL FOREIGN KEY REFERENCES Customer(CompanyName_PK),
        PriorityLevel_PK tinyint NOT NULL,
        ResponseHours int NOT NULL

        CONSTRAINT CustomerPriority_PK PRIMARY KEY CLUSTERED(CustomerID_PK, PriorityLevel_PK)
)

CREATE TABLE DefaultPriority (
        PriorityLevel_PK tinyint PRIMARY KEY,
        ResponseHours int NOT NULL,
        SolveHours int NOT NULL
)



/* INSERT SYSTEM DATA */
```

```
INSERT INTO Role VALUES('T', 'Technician')
INSERT INTO Role VALUES('A', 'Administrator')
INSERT INTO Role VALUES('M', 'Office Manager')

INSERT INTO CallStatus VALUES('A', 'Authorised')
INSERT INTO CallStatus VALUES('C', 'Complete')
INSERT INTO CallStatus VALUES('I', 'In Progress')
INSERT INTO CallStatus VALUES('P', 'Awaiting Closure')
INSERT INTO CallStatus VALUES('R', 'Rejected')
INSERT INTO CallStatus VALUES('W', 'Awaiting Authorisation')

INSERT INTO ProblemCategory VALUES('D', 'Duplicate Problem')
INSERT INTO ProblemCategory VALUES('H', 'Hardware Problem')
INSERT INTO ProblemCategory VALUES('S', 'Software Problem')
INSERT INTO ProblemCategory VALUES('U', 'User Problem')
INSERT INTO ProblemCategory VALUES('X', 'Unknown')


INSERT INTO CallPriority VALUES(1, 'Critical')
INSERT INTO CallPriority VALUES(2, 'Inconvenient')
INSERT INTO CallPriority VALUES(3, 'Low')
```

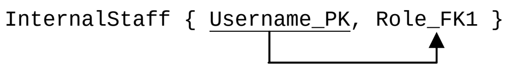
Appendix L – Database functional dependencies



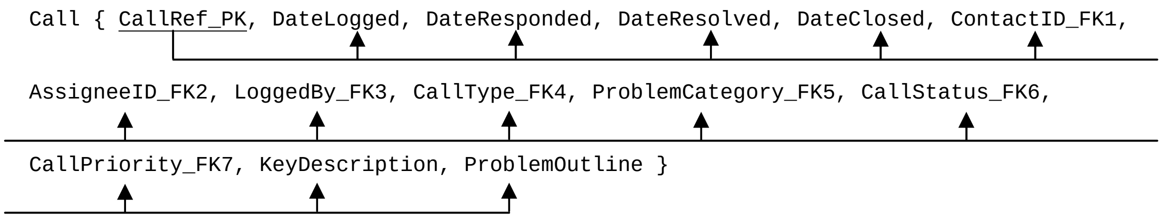
Customer { CompanyName_PK, Address1, Address2, Address3, Address4, Postcode, Telephone, Fax }



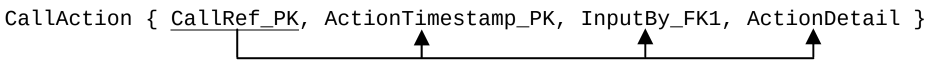
CustomerEmployee { CustEmpID_PK, Customer_FK, FirstName, Surname, Initial, JobTitle, emailAddress, Telephone }



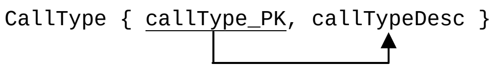
InternalStaff { Username_PK, Role_FK1 }



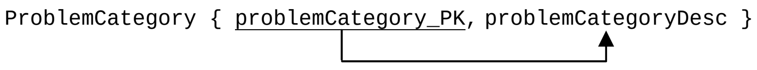
Call { CallRef_PK, DateLogged, DateResponded, DateResolved, DateClosed, ContactID_FK1, AssigneeID_FK2, LoggedBy_FK3, CallType_FK4, ProblemCategory_FK5, CallStatus_FK6, CallPriority_FK7, KeyDescription, ProblemOutline }



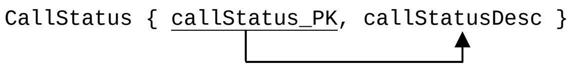
CallAction { CallRef_PK, ActionTimestamp_PK, InputBy_FK1, ActionDetail }



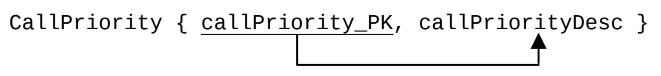
CallType { callType_PK, callTypeDesc }

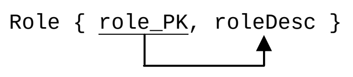


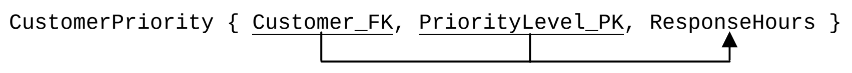
ProblemCategory { problemCategory_PK, problemCategoryDesc }

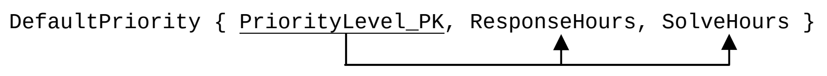


CallStatus { callStatus_PK, callStatusDesc }

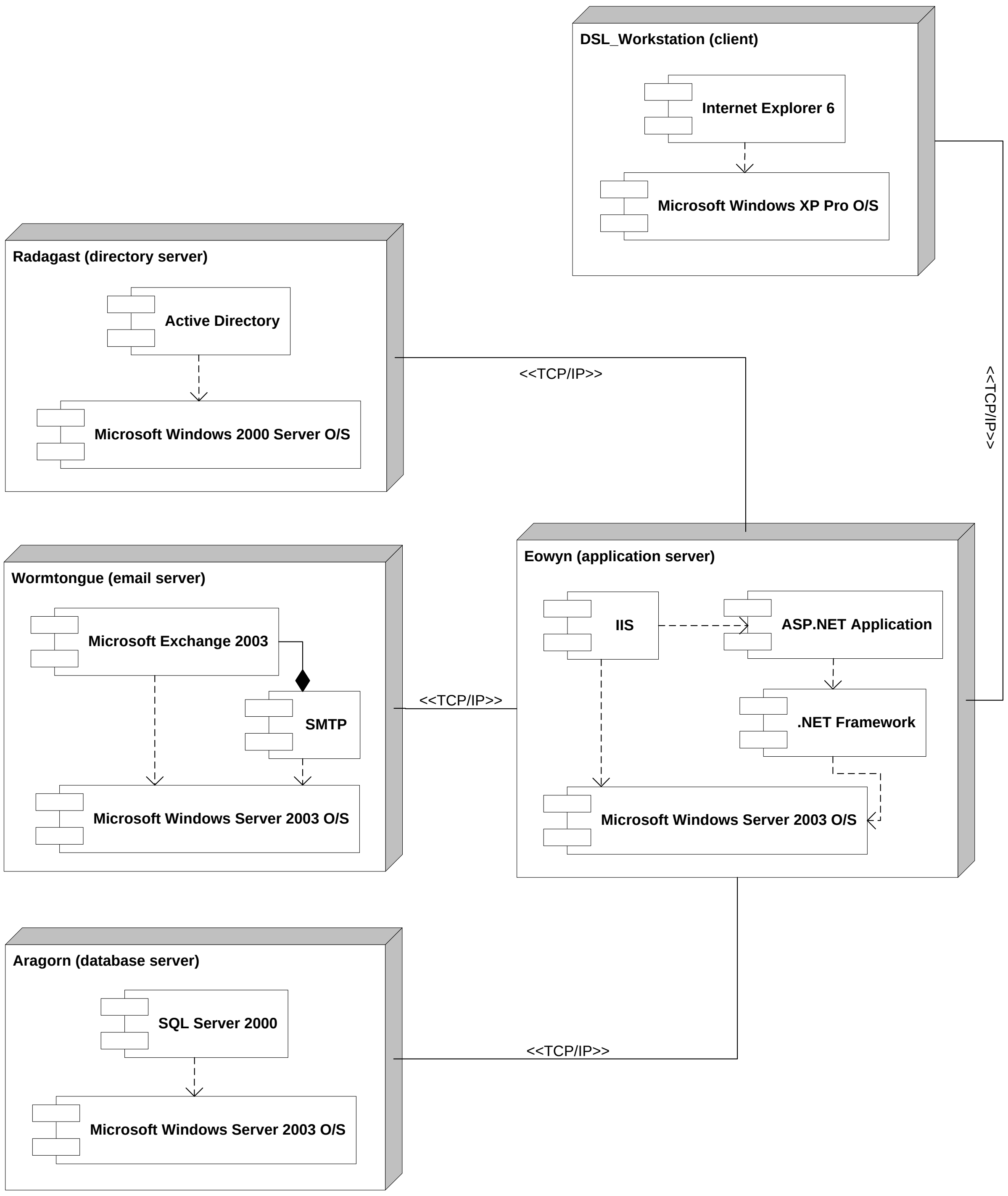

CallPriority { callPriority_PK, callPriorityDesc }


Role { role_PK, roleDesc }


CustomerPriority { Customer_FK, PriorityLevel_PK, ResponseHours }

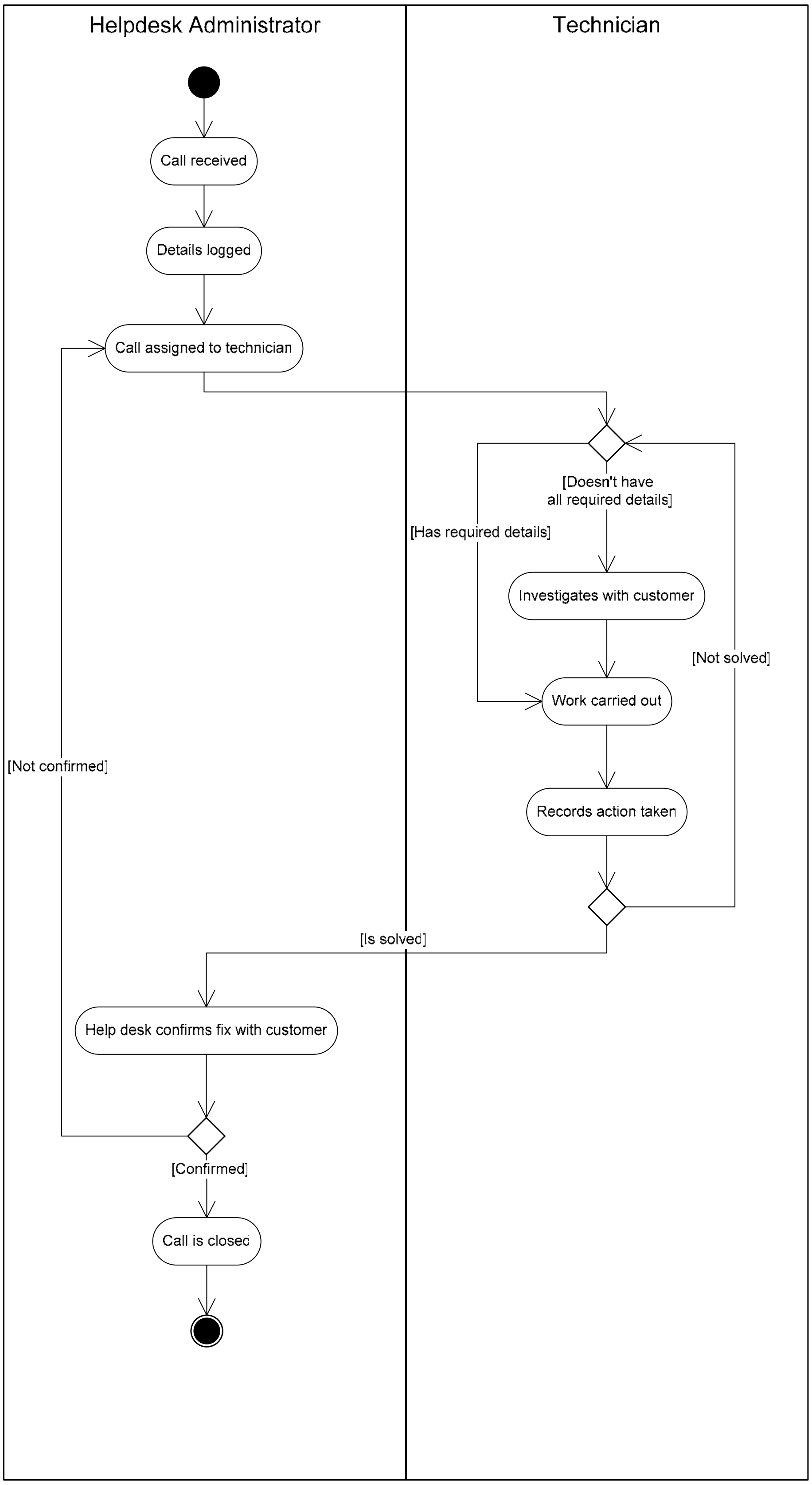

DefaultPriority { PriorityLevel_PK, ResponseHours, SolveHours }

Appendix M – System Architecture


DSL_Workstation (client)
Internet Explorer 6
Microsoft Windows XP Pro O/S
Radagast (directory server)
Active Directory
Microsoft Windows 2000 Server O/S
<<TCP/IP>>
<<TCP/IP>>
Wormtongue (email server)
Microsoft Exchange 2003
SMTP
Microsoft Windows Server 2003 O/S
<<TCP/IP>>
Eowyn (application server)
IIS
ASP.NET Application
.NET Framework
Microsoft Windows Server 2003 O/S
Aragorn (database server)
SQL Server 2000
Microsoft Windows Server 2003 O/S
<<TCP/IP>>

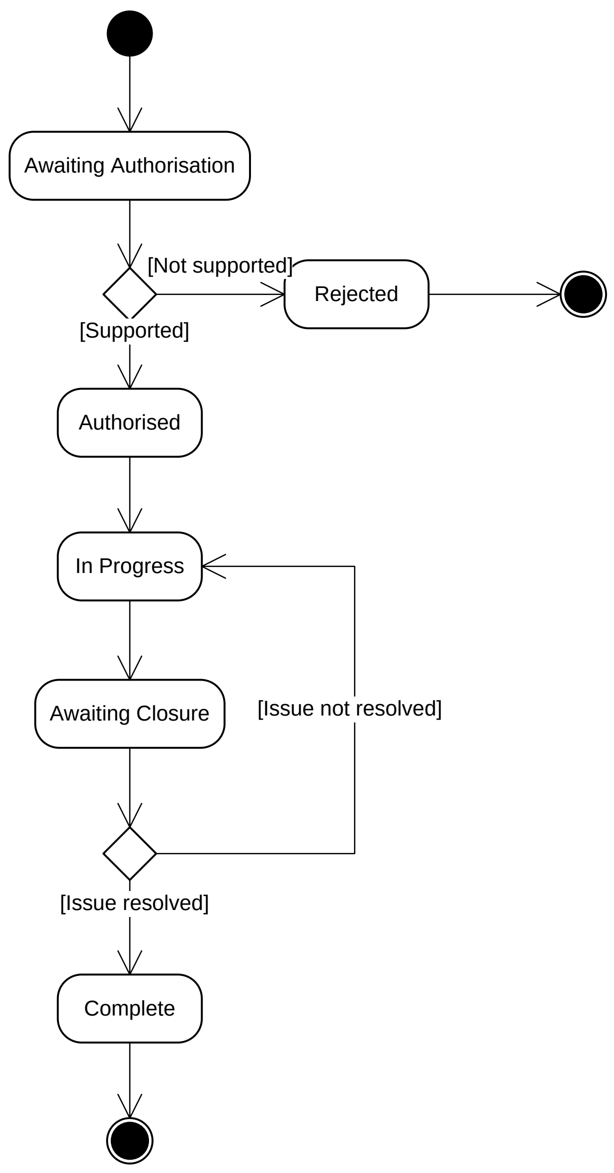
UML Implementation diagram specifying the new system architecture

Appendix N – Support call process



UML Activity diagram showing the support call process.

Appendix O – Support call status’



UML Statechart displaying call status’

Appendix P – Functional Test Execution Document

Task	Description	Success?	Accepted
1.1.1.	Log a new call		Y
	Successfully complete task	Y	
	Insert special characters into 'Brief Description' field	Y	
	Insert special characters into 'Problem Outline' field	Y	
	Successfully emails assignee the call details	Y	

Task	Description	Success?	Accepted?
1.1.2.	Update support call log		Y
	Successfully complete task	Y	
	Insert special characters into 'Add detail to call' field	Y	

Task	Description	Success?	Accepted?
1.1.3.	Obtain status of call		
	Successfully obtain the status of a call	Y	

Task	Description	Success?	Accepted?
1.2.1.	Add customer to list		Y
	Successfully add a customer to the list	Y	
	Insert special characters into 'Company Name' field	Y	
	Insert special characters into 'Address 1' field	Y	
	Insert special characters into 'Address 2' field	Y	
	Insert special characters into 'Address 3' field	Y	
	Insert special characters into 'Address 4' field	Y	
	Insert special characters into 'Postcode' field	Y	
	Insert special characters into 'Telephone' field	Y	
	Insert special characters into 'Fax' field	Y	
	'Telephone' field accepts only numbers?	N	
	'Fax' field accepts only numbers?	N	
	'Postcode' field is properly validated	N	

Task	Description	Success?	Accepted?
1.2.2.	Display all customers		Y
	Successfully display all the users	Y	

Task	Description	Success?	Accepted?
1.2.3.	Display customer details		Y
	Successfully display all the customers details	Y	

Task	Description	Success?	Accepted?
1.2.4.	Update customer details		Y
	Successfully update the details of the customer	Y	
	Fields are validated	N	
	Insert special characters into fields	Y	

Task	Description	Success?	Accepted?
1.2.5.	Enter SLA details		Y
	Successfully change SLA from system default to custom	Y	
	Successfully change SLA from custom to system default	Y	
	Validate for only numbers	N	

Acceptable for first prototype

Task	Description	Success?	Accepted?
1.2.6.	Add / Remove client employee		Y
	Successfully add an employee against a client	Y	
	Successfully remove a client from an employee	N	
	Can insert special characters into employee name	Y	

Acceptable for first prototype

Task	Description	Success?	Accepted?
1.3.1.	Add / Remove system users		Y
	Successfully add an user onto the system	Y	
	Successfully remove a user from the system	N	

Acceptable for first prototype

Task	Description	Success?	Accepted?
1.4.1.	View / Update default SLA response times		Y
	Can view and update the system default SLA response times	N	

Hard coded variables are acceptable for first prototype

Task	Description	Success?	Accepted?
2.1.1.	View assigned calls		Y
	Can view calls assigned to member logged on	Y	

Task	**Description**	**Success?**	**Accepted?**
2.1.2.	Update call log with action taken		Y
	Can successfully log action taken to the call	Y	
	Field accepts special characters	Y	
	Can alter status and assignee	Y	
	Can email assignee when updated	N	

Acceptable for first prototype – email functionality has been demonstrated elsewhere

Task	**Description**	**Success?**	**Accepted?**
2.1.3.	View customer contract details		Y
	Can view and the contract details	Y	

Task	**Description**	**Success?**	**Accepted?**
2.1.4.	Pass call to third party		N
	Can successfully pass he call to a third party	N	

Does not form any of the minimum requirements agreed. Accept that this is a future development

Task	**Description**	**Success?**	**Accepted?**
3.1.1.	Chase technicians that have not updated an open call in a week		Y
	Successfully obtain a list of technicians not updated call in the week	N	
	Can email technicians who have not updated their calls in the week	Y	

Email is acceptable for first prototype though it would be nice to view a list in the actual system

Task	**Description**	**Success?**	**Accepted?**
3.1.2.	Beware of calls breaching / about to breach SLA		Y
	Is able to obtain list of calls breaching SLA	N	

Email is acceptable for first prototype though it would be nice to view a list in the actual system

Task	**Description**	**Success?**	**Accepted?**
3.1.3.	Raise priority of a call		Y
	Can raise the priority of the call	Y	

Task	**Description**	**Success?**	**Accepted?**
3.1.4.	Get statistics out of the system		Y
	Can successfully obtain statistics from the system	N	

Does not form any of the minimum requirements agreed. Accept that this is a future development

Task	**Description**	**Success?**	**Accepted?**
4.1.	Email Support call details to a system user		Y
	Functionality to email support call details to a system user	Y	

Task	**Description**	**Success?**	**Accepted?**
4.2.	Email support technician when about to break SLA		Y
	Email engine implemented and working	Y	

Appendix Q – Usability Testing Data

Julie		Attempt 1	Attempt 2	Attempt 3
	1	00:52	00:40	00:28
	2	00:27	00:34	00:18
	3	01:10	00:55	00:54
	4	00:29	00:47	00:17
Paul				
	1	01:45	00:26	00:29
	2	00:59	00:29	00:19
	3	03:17	00:50	00:40
	4	00:52	00:45	00:18
Barry				
	1	00:00:57	00:00:52	00:00:29
	2	00:00:27	00:00:23	00:00:23
	3	00:01:42	00:01:29	00:01:27
	4	00:00:28	00:00:24	00:00:23
Ramsay				
	1	00:01:25	00:00:47	00:00:38
	2	00:00:46	00:42	00:21
	3	00:01:25	01:03	01:02
	4	00:00:32	00:35	00:29

Task 1			
Min	00:52	00:26	00:28
Max	01:45	00:52	00:38
Average	01:14.7	00:41.3	00:31.0
Task 2			
Min	00:27	00:23	00:18
Max	00:59	00:42	00:23
Average	00:39.7	00:32.0	00:20.3
Task 3			
Min	01:10	00:50	00:40
Max	03:17	01:29	01:27
Average	01:53.5	01:04.3	01:00.7
Task 4			
Min	00:28	00:24	00:17
Max	00:52	00:47	00:29
Average	00:35.2	00:37.7	00:21.8