Regis University
ePublications at Regis University

All Regis University Theses

Fall 2007

Application Packaging Tracking System

Cormac Quillinan
Regis University

Follow this and additional works at: https://epublications.regis.edu/theses

Part of the Computer Sciences Commons

Recommended Citation

Regis University
College for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Application Packaging Tracking System

By: Cormac Quillinan

Regis University

School for Professional Studies

Master of Science in Software and Information Systems

Prepared by: Cormac Quillinan	Creation Date: 10th November 2006
Approved by:	Mark S. James
Version	1.0

Status	Version No.	Author(s)	Date	Remarks
Draft	0.1	Cormac Quillinan	10/11/06	Document creation
Draft	0.2	Mark James	29/06/07	First full draft review. Corrections and suggestions made
Draft	0.3	Cormac Quillinan	09/07/07	Suggested corrections made to document
Draft	0.4	Cormac Quillinan	20/07/07	Document work in progress
Draft	0.5	Mark James	22/08/07	Corrected mistakes in chapter 1 and 2
Draft	0.6	Cormac Quillinan	23/08/07	Revised Chapter 1 and 2
Draft	0.7	Cormac Quillinan	24/08/07	Revised Chapter 1, 5 and 6
Draft	0.8	Cormac Quillinan	25/08/07	Reviewed all chapters and corrected any grammatical errors found.
Draft	0.9	Cormac Quillinan	26/08/07	Chapter 6 and 1 were updated. Abstract was written.
Approved	1.0	Cormac Quillinan	30/08/07	Submitted for approval and grading

Abstract

In the business world it is very difficult to manage and track user application installations and licensing. The primary problem with this area is currently there is no single solution that encapsulates the entire process. The current commercially available tracking systems, such as SMS, DevTrack and Installshield Tracking system, do not provide a complete solution. These products provide partial solutions. However, they do not address the entire problem. In this thesis we proposed an Application Packaging Tracking System. This system tracks application installations throughout the entire process, from submission to deployment.

Acknowledgements

This master thesis has been a difficult and challenging experience during these last two years. There were many different people without whose help and I could not have completed this. I would like to sincerely thank my brother Thomas, for his guidance and support.

I would like to thank my family, especially my parents who have been extremely supportive and patient during this master's program. Finally I would like to thank Mark James for his ongoing support in the development of this thesis. He has been very patient and very quick to respond with corrections and comments.

Table of Contents

List of Tables

List of Figures

Chapters

1   Introduction

Application packaging, migration and inventory are an established necessity in large managed corporations. If it has not already been outlined or highlighted as an Information Technology (IT) overhead in a given business, it will in the future. Given that this reality will be important to understand how many applications business uses; the licenses they have, the application users; and so on. It soon becomes apparent that a tracking system to efficiently manage this process is needed. If a given company does not already have such a system, then the information managed could potentially be in breach of licensing agreements and they can be fined by any and possibly all of software vendors that a company works with.

> "At the BSA we understand how easy it is to let software management slip. As a result, many businesses today still breach copyright laws, despite penalties including fines of up to EUR127,000, five years in prison, or both."

> "We've designed the Anti-Piracy Information section to help users prevent software theft. BSA — one of the World's leading anti-software piracy groups — is committed to providing support every step of the way. In 2003, across the EMEA region, the BSA handled 57,625 calls, followed up 7,929 end user leads and took legal action against 9142 companies."

[Software Asset Review 2006 and Anti-Piracy Information Retrieved on 26 August 2007 from http://w3.bsa.org/ireland/]

To avoid these types of situations it has been proposed to develop an Application Packaging Tracking System that will be used to track applications in the business from the submission stage right through to deployment. We will examine each of these stages in turn in this chapter.

1.1   Problem Statement

One problem encountered with big corporations is software tracking, people have both necessary and unnecessary software installed on their computers.   The employees computers become overloaded with these applications, the IT staff are not completely aware of who has what software and how many licenses are being used currently. This has become a huge problem and in recent years software vendors are becoming increasingly stringent about software licenses in businesses.

> "According to a recent study by IDC, 67% of companies do not track software usage. Without accurate software usage data, organizations routinely overspend and buy licenses they don't use, or under spend and deny end users access to the software they need. They also waste money renewing licenses that never get used." [Tracking Software - Retrieved on the 21st September 2006 from http://www.macrovision.com/solutions/it/tracking/index.shtml]

A basic requirement of a planned migration of applications and operating systems is a tracking management system. Imagine a financial enterprise that are migrating an application and Operating System (OS). They have decided to move the business from Windows NT to Windows XP. Another basic requirement is that all of the applications that each department uses are available, and installed, on this new system.   One basic problem that arises is no one in the business knows what applications are required for each department, how they are install and so on. The lack of such a tracking system will result in the failure of such management and migration projects.

Application packaging/deployment process
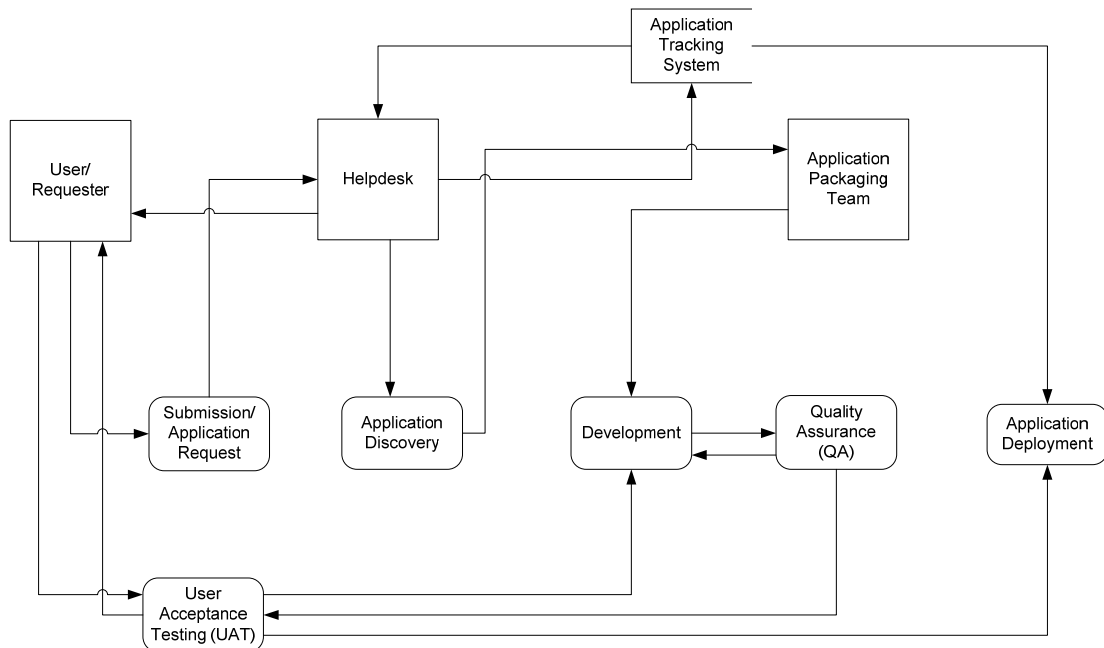
Figure I Application packaging and deployment process

Again we can use our business scenario. The process diagram shown in Figure I is the proposed solution for user application requests for their department, or for their own personal computer. With the help of this proposed system, we can track an application deployment project from the submission/request phase through to the development and deployment phase. This involves the user or requester sending a request to the IT helpdesk. At this stage, they will query the application tracking system to see if the application has already been developed and ready for deployment. If the application is ready, it will be immediately deployed to the user. Otherwise, it will then be entered into the system as an active application request.

Once the application has been entered in as a submitted request, it is at the "Application Discovery" phase. This includes the installation details and documents with any customisations required for an application to be installed as per the user's

request. This information is gathered by the helpdesk from the user and then entered into the system. It will then be passed to a developer for application development.

The developers in the *Application Packaging Team* will then review the application request and make expert decisions on how it will be developed and deployed with the information gathered. There are currently two methods used for deployment in this business: either through Softricity SoftGrid Applications or Microsoft Windows Installer (MSI) wrapped with System Management Server (SMS) deployment. These will be discussed in more detail in Chapter 2. After the developer's initial review, they will then provide an estimated time of arrival (ETA) to the user, to ensure communication is established and a good relationship will be built up between developer and user for further stages such as user acceptance testing (UAT).

Quality Assurance (QA) is the next phase of the process. On completion, the application is sent by the developer to a team of QA testers, who will then decide if the package meets the standards, user requirements and if the application is ready for delivery to the user. If the application passes QA it will then be sent to the user for acceptance testing.

UAT is important because developers cannot test applications to the same level as the users themselves. The users are then contacted to perform UAT and approve the application in question. Once the applications have completed all the previous processes, we need to track the UAT and provide an approval at this point to allow the application to go to deployment. If the user has signed off and are completely satisfied with the application it will continue to application deployment. However if it does not

meet the standards it will return to the development stage and be updated until it meets the user requirements.

The final phase is "Application Deployment". This phase tracks the application name and version that has been approved, and sends it for deployment to the user's computer. The system will be updated to show that the application is in production and can be used on request. These steps are vital to allow the completion of a successful migration of each individual application.

1.2   Thesis Statement

The aim of this thesis is to provide a management and migration architecture for the financial services industry. A system of this type relies on a well structured and managed Application Packaging Tracking System. This project is intended to provide an integrated tracking system that will allow users to successfully track and manage their applications from submission to acceptance.

1.3   Requirements Document

There are two types of requirements that one must consider when organising and understanding the system that is required.   These are the functional and non-functional requirements.

The functional requirements are concerned with the basic services that the system is expected to provide. This concerns how the system will react to certain inputs and how the system should behave in particular situations. The non-functional requirements are concerned with the constraints on the services or functions offered by the system. Non-functional requirements may relate to system properties such as response time, reliability and security.

When investigating on the requirements for the Application Packaging Tracking System, significant requirement analysis was necessary.   This gives a better understanding of who would be using the system, what the current situation was within the application deployment departments, when must the new system be put into use, and how will the new system function.

The principle roles involved in this system are the current employees, development team and the helpdesk. Employees, development team and the helpdesk will have different roles within the organisation. The role of the development team is to work with other employees and the departments to get an understanding of the applications they require in order to function effectively. The role of the employees is to enter this data into the system. Helpdesk, or as they are often called, "Desktop Support" will have a major role within the system as they have a complete overview of the system

and, therefore, an understanding of what applications are required, the departments they are required for, and how they are deployed. It will also provide a means to track changes in applications and employees within the business.

We refer back to the scenario mentioned in Section 1.1 where, a financial corporation has an application and OS migration project. Presently, no current system addresses their requirements. We will discuss existing solutions in more detail in Chapter 2.

There are many different excel spreadsheets capturing the applications per department. Typically many of these audit logs are outdated. Such logs also include lists of users per department. The helpdesk are hindered with calls regarding user applications for new users and it is a difficult task to discover what applications are in production; whether automotive deployment is possible or instead a manual install of the requested application onto the user's computer will be required.

1.3.1   Functional Requirements

Using the financial scenario, we have already introduced a number of different actors that will be involved in the process. We have users of the system, employees, helpdesk staff, management, administrators and developers.

Users of the system should be able to view, add, update and delete all application details. They will also be responsible for adding discovery information for each individual application requested. Another requirement is that users should be able to see what applications are in use by each department. There is also an overall understanding that all users should be able to login securely to the system.

The developers must be able to update information stored on the database such as vendors, languages, active directory groups, and developers. Both developers and administrators should be able to create and modify any application specific information, including the process and status.

Helpdesk users should be allowed to add new users into the system, update and add department information. On a monthly basis, management staff should be able to generate a wide range of reports from the system. Such reports might include a list of current applications in the system, list of current users, what applications are in development and what applications are in production.

1.3.2   Non-Functional Requirements

The non-functional requirements are in a number of disjointed areas. These areas are the non-functional requirements into the user interface, documentation, and error handling. Using the same actors introduced above, we can now breakdown these specific areas, and describe the non-functional requirements for each of these actors.

User Interface

Users will not be knowledgeable of the new system and, therefore, there will be a need for training and reference documentation for the system. The system will cater to more than one type of user. The system should be easy to learn and should be intuitive, as most usage will be intermittent.

Errors should be minimised to enhance ease of use, and also to increase user satisfaction. The only input device will be either the keyboard, or scanner, in order to scan images or documents.

Documentation:

Reports such as monthly applications in production, UAT or QA will be required from the system. Management will request these reports, so these must be easily retrievable and give specific important information.

Error Handling and Extreme Conditions:

The system must be user friendly so any and all messages should be understandable and have a good meaningful response. If system is asked by the user to process a file larger than the system capacity an error message should be provided to the user. User input should always be checked for validity and error messages should be provided. Users should also get the chance to review entered data.

1.4   Contribution

This project solution is a proof of concept. Once we have investigated and proved that this system is a good solution for this type of business, we can then provide an Application Packaging Tracking System that can be used to properly document and ensure correct processes and procedures are followed. It will be able to show a clear path to what applications are managed to what departments and going forward we can have a more structured approach.

Some of the benefits should include; new users tracked correctly in the system, all applications developed tracked new and old; applications owned and used per department will now be tracked and there will be no need for department IT audits. The proposed system should be OS independent. This will allow us to integrate this system into any business environment. The system should be updatable and display information for each specific application. For example, it should give information how an application can be deployed, the users that already have this application deployed, what developer worked on this application, and so on.

 It should be clear that this solution will reduce costs and time. We refer to the scenario introduced earlier. The business is missing an application. The relevant department calls the development team. The development team has a manual process of searching for installation documents, UAT users, and development information and so on. This will allow the development team and the helpdesk to be more efficient in their daily duties. If another migration project like this is proposed, it would then be a simple task to track and develop the necessary improvements to the current applications and users in the system.

1.5   Organisation of this dissertation

This dissertation will outline the necessary documentation and steps taken to develop the Application Packaging Tracking System as a proof of concept for the scenario introduced earlier in this chapter. We will also understand from the later chapters how the project began and how we came to the final result.

In Chapter 2, research into this project is introduced. We will begin by discussing the background of this project and type of system. In order to understand the other systems that are available and why they cannot meet the requirements outlined, we will summarise existing products in this area. This will then lead into a discussion on the research methods.

Once we have established the relevant research areas in Chapter 2, we will then focus on methodologies in Chapter 3. This includes both the deliverables that are required of the project and the intended result once the project has been fully developed.

We will then discuss the project software life cycle that will be used for development. We will then explore the software models, where UML will be introduced as a method of design and development with the use of Use Case diagrams and Class diagrams. We will also discuss database design at that time.

The review of the deliverables will be introduced next. It will be necessary to revisit the goals of the project. This will lead into a discussion on quality control and conclude with the desired results.

Chapter 4 is the design and implementation chapter. It will review the project development. We will then discuss the project management and schedule, the milestones and any plan modifications. The database design becomes necessary here, we will detail the design of table structure and database catalog. An entity relationship diagram will be drawn up and shown to detail the structure of the database.

Software implementation is then introduced. We will begin designing the code by using the UML design diagrams discussed in Chapter 3. We will use two separate diagram designs in this section, use case diagrams and class diagrams. We will then discuss the Graphical User Interface (GUI) front-end design. Here we will design a GUI prototype that will be used to develop the system. This will give us a valuable understanding of the requirements and will give us a better understanding of what can be expected from the system.

Once we have concluded the discussion about the design and implementation decisions, we can then proceed to Chapter 5, where we will analyse the results. We will evaluate the system design and development. We will then discuss the issues and limitations that arose throughout the project timeline, and discuss the benefits of the project due to its current design and implementation. The chapter will then proceed to discuss the impacts occurred during the project life cycle and finally conclude with the findings and analysis of this system.

Chapter 6 will be the conclusion. Here we will discuss what have we learned from the development of this project and what would be done differently, now that we have a better understanding. Here we will give a brief summary of certain aspects that could

have been approached differently. We will then summarise the project solution, development and the documentation of this system.

This chapter will then focus on the project expectations and the next steps if the project were to continue. We will conclude with some suggestions of future work.

2   Research

2.1   Background:

We have already introduced some deployment and migration terms and methods in Chapter 1.   In this chapter, we will expand on these concepts and develop an understanding about the requirements for the application tracking system.   There are many existing products that involve the migration of applications and operation systems (OS) from old versions of Windows NT to Windows 2000, or to Windows XP.   In the near future, as market acceptance grows, Windows Vista will become more widely used. This will again increase the need of application migration.   In this chapter, we will again us this scenario introduced in Chapter 1.   Some in-house developed applications are no longer supported, or no one in the business understands how to install them or how to rebuild these applications.

We will first take a broader view at Microsoft Windows Installer (MSI) technology. We will then proceed to discuss other types of application development and deployment technologies. This includes application virtualisation, manual installations and Microsoft Systems Management Server (SMS) as a deployment tool.

2.2   Research Overview:

We have briefly mentioned some of the technologies needed and used for developing and delivering applications to the users desktop.   These include MSI, SoftGrid application virtualisation and manual installations.   We will expand on these technologies and give a broader overview of how the applications are developed.

Once we have introduced these technologies we will explore how the toolsets are used to develop the applications and how these will integrate into the application tracking tool.   We have previously discussed the Application Packaging and deployment process in Chapter 1. Ideally this would form core of a complete application deployment and delivery implementation.

2.3   Research Methods:

In this section, we will discuss the research methods that will be used to develop a better understanding of the system requirements and how we will approach the developing process.

Solutions currently exist in this problem area. However, these products or solutions do not provide the necessary level of detail required for an application tracking system. It is the intention to research the existing market projects and develop an understanding of their core components.

Such products that are available are

- DevTrack - http://www.hallogram.com/tedevtrack/index.html

- Installshield - http://www.macrovision.com/solutions/it/tracking/index.shtml

Both the back-end and visual interface of the product must be defined and developed. Defining the backend database (DB) and the development of the client graphical user interface (GUI) depends on both the requirements of users and the functional capabilities needed.

The demand for high level change management systems and application tracking system is increasing. There is a need to understand what is required to create this system from a backend DB perspective and also a front end GUI. The look and feel of the system is as important as the functionality and the backend DB structure. The application must not only be visually appealing but also fulfil the requirements documented during the research phase.

There are a vast number of companies that have, and are, currently migrating and updating their systems. These companies are an excellent source of information that can be accessed through personal and business contacts. This method of investigative research has proved to be invaluable. Interviews can be used to discuss the end to end workflow process of these companies, to gain insight from both a high level and a low level perspective.

2.4   Research Technologies

Microsoft Windows Installer (MSI) is a method of customisation application installations for specific user requirements.   The first MSI project available was Microsoft Office 2000.   A MSI is an installation package that is created from a generic setup. Commonly vendors supply installation using MSI technology.

> "An installation package contains all of the information that the Windows Installer requires to install or uninstall an application or product and to run the setup user interface. Each installation package includes an .msi file, containing an installation database, a summary information stream, and data streams for various parts of the installation. The .msi file can also contain one or more transforms, internal source files, and external source files or cabinet files required by the installation."

[Installation   Package   –   Retrieved   on   the   06th   May   2007   from http://msdn2.microsoft.com/en-us/library/aa369294.aspx]

MSI are used to reduce the installation overhead and to increase the capabilities of installers.

> "The Windows Installer reduces the total cost of ownership (TCO) for your customers by enabling them to efficiently install and configure your products and applications. The installer can also provide your product with new capabilities to advertise features without installing them, to install products on demand, and to add user customizations."

[Overview of Windows Installer – Retrieved on the 06th May 2007 from http://msdn2.microsoft.com/en-us/library/aa370566.aspx]

Microsoft Systems Management Server (SMS) is a method used to deliver applications that have been developed in MSI to the users desktop.

"Installing and maintaining software is a major cost to corporations with locations across a wide geographical area. In fact, most of the cost of maintaining a corporate computer system comes from the software installation, support, and maintenance—not from the initial licensing of the software itself."

"Systems Management Server enables you to centrally manage your entire organization. Using Systems Management Server, you can distribute and install software on clients and servers across your corporate network, set up Systems Management Server network applications, automatically collect and maintain hardware and software inventory, provide direct support to clients, and monitor your network."

[Chapter 1 - Introducing Systems Management Server.

Retrieved   on   the   10th   August   2007   from http://www.microsoft.com/technet/prodtechnol/sms/sms2/proddocs/smsplan/smsplan.mspx?mfr=true]

SMS helps us in the deployed of MSIs but we can also use its more powerful systems to maintain the hardware inventory.

A recent development is a new method of application deployment that is called SoftGrid Application Virtualization. This was originally developed by a company called Softricity. Softricity was recently acquired by Microsoft, and has now become their central focus for application delivery.

We have already introduced the concept of application virtualisation. This is a relevantly recent development. Softricity was one of the first to market with this concept. Microsoft also determined that this is the future of application deployment, so they quickly made a bid for Softricity acquired them. The product has been renamed Microsoft SoftGrid Application Virtualisation. Application virtualisation is the process by which applications are install remotely yet it appears to the user the applications are installed locally.

> "Enables applications to run without the need to ever visit a desktop, laptop or server. Applications are no longer installed on the client – and there is absolutely no impact on the host operating system or other applications. The most extensive virtualization technology on the market, it virtualizes per user, per application instance, every key application component, as well as Windows Services. As a result, application conflicts, and the need for regression testing, are totally eliminated."

[The SoftGrid Application Virtualization Platform – Retrieved on the 04th April 2007 from http://www.softricity.com/products/softgrid.asp].

Not only does it manages the user license, and is a user based functionality, there is never a need for the user to have applications installed on their local machines. Users can move from machine to machine on the network and their applications will move with them. If they are laptop users they can roam offline and still use these applications. This operates as follows. Once the users have initially logged onto the system and launched their applications. The application will cache the required files on a virtual drive that remains even after you disconnect or reboot your computer. If

the user has not launched an application while connected to the network, this application will display an error. However, if the user can connect to their work systems via the Internet, or even connect directly, and the application will then be launched. This allows users to roam and launch their applications offline.

The final method of application deployment is manual installation. It is generally understood that to redevelop applications into either SoftGrid or MSI takes some time and resources. Given that there are costs involved in achieving this, we have to balance whether it is feasible or even achievable, to deliver the application using the proposed method. For example, if we have a single user in the whole business that requires a certain application, and this application will never be required by anyone else, is it feasible to spend time or money on redeveloping it as a managed deployment? Typically in this case, we would document the installation and perform it manually.

We have discussed the process area and methods of application installation and deployment. However, there are existing commercial tracking tools available. SMS has been mentioned earlier as an application management and deployment tool. SMS tracks local MSI and executable applications. We can query machines on the system and get a list of applications on each system. SMS can manage users desktop and give us a list of users that have logged on in last 14 days. However, it lacks the full process cycle; it does not yet manage SoftGrid Virtualization applications or Critix applications. It also cannot properly manage users in a system and does not give an accurate query of users or departments in the business. We cannot store installation documents or even track an application progress from submission to deployment.

SMS only provides part of the process required to properly manage the application packaging and deployment process.

Other tools exist that more directly address tracking, such as Devtrack.

"DevTrack is the premiere defect- and project-tracking tool designed specifically for software development teams. DevTrack comprehensively tracks and manages all defect information, change requests, feature enhancements, project related documents, HTML links, and other development Issues."

[DevTrack   Retrieved   on   the 12th September   2006   from http://www.hallogram.com/tedevtrack/index.html]

We can see from this that DevTrack is more a project tracking system. It does not specifically address the entire problem area. It can track time and project, but cannot properly integrate with the application packaging process.

Installshield and Wise Solutions also have application tracking tools that specifically address the packaging process. However, these tracking tools are specific to MSI deployment only. Even if they did incorporate manual installations, Critix applications and SoftGrid application virtualisation, these tools would only be able to manage the development part of the process. All of these are very powerful toolsets but are all very specific to their own area. The Application Packaging Tracking System will cover all the application packaging and deployment process areas. It will not focus on one area but it will include all of the current and missing modules. None of the currently proposed commercial solutions track users or business departments.

Recall the business scenario introduced in Chapter 1. As there has been a need for software applications to be delivered to a computer in three different methods, it is necessary for the developers, helpdesk and support personal to understand what applications are required; how can they be delivered; and what is available in each deployment method. These are all key reasons that indicate this type of software tool is necessary. At present there is no such solution. This information must be obtained manually by contacting each department, and each of the users to discover their application requirements, and then check our systems to see if the applications are available. This information is tracked in a excel spreadsheet. This spreadsheet often becomes lost, outdated and sometimes even obsolete.

It is readily apparent that this is a poor approach to tracking applications. The application tracking system is not only important for the daily tasks of both the helpdesk and the development team, but to the more complicated task of OS migration. If this system were available then we would be able to track and list all the current applications in the system, find out whether or not they will function in this environment. If they need to be redeveloped, all the information would be available through the system

If we imagine that a business using the Application Packaging Tracking System. If the business now decides to move to Windows Vista. As the hard work has already been completed, the applications are correctly managed and so on. It should now only be a matter of testing these applications on the new business OS build, in order to certify these applications and then update the tracking system accordingly.

We have discussed the application packaging methods, but we need to understand that we also will have an application packaging toolset to create and deliver these applications in these methods.   There are several different products available for creating and configuring MSIs. The two most common tools are Wise for Windows Installer and FlexNet Installshield Adminstudio. These tools help create and deliver the MSI to users. The other major toolset is, of course, application virtualisation using Microsoft SoftGrid. There are other application virtualisation products such as LANDesk and ThinInstall, but Microsoft is currently the market leader.

2.5   Internationalisation

We need to understand that all of these products have internationalisation in mind. It is of this understanding that the future around application packaging and delivery have moved from a localised site specific delivery to a global mindset. With this in mind, we also include the requirement that the management tool must correctly handle internationalisation of applications.

2.6   Testing and Deployment

We have already discussed how we develop and customise application installs for the business, but what we have not discussed is the QA, UAT and deployment processes. QA (Quality Assurance) is performed internally by the development team.   The developer of an application install must perform QA testing of their own work, and also pass it to another member of the team to follow a QA test plan. At this stage, if any flaws are found or standards are not met, the application will go back to the developer for rework until all the flaws are corrected. The application is then passed back to QA for approval.

Once the application has successfully passed the QA process, the user or application requester is contacted. It is necessary for them to test the application in question to see if it meets their requirements and expectation.   Typically, an independent computer is set up in a test environment and the application is installed using the SMS method if it is an MSI, or if it is a SoftGrid application it is assigned to the user's account. The user will then login with user privileges and test normal functionality. If the user is not satisfied with the testing they fill out a form detailing where the problems exist and it returns to the development team for rework and retesting. If the application is approved, the user sends an approval form to development team.

Once the application is approved it is then moved into the approved area, and global use is permitted. The system is then updated to allow the application to be deployed.

3   Methodology

In this chapter, we consider the methodology and we examine the requirements of this project. We will begin with what the deliverables will be. We will then proceed to discuss the project life cycle. Then we discussed the software models that are required. The software models are a large discussion topic. Brief descriptions will be given here to develop a better understanding of the process we are required to follow. We will then discuss quality control and finally we will introduce the goals and outcomes that we hope to achieve.

3.1   Deliverables

This project is a proof of concept, if at the end of this dissertation it has been proven to be a successful delivery it will then be consider whether or not to proceed with the full development and deployment of this "Application Packaging Tracking System" We are hoping that this product provides the "missing link" in management tools for the software packaging deployment process.

If this project is successful we hope it will determine the key factors for a successful migration and have the ability to adapt to the requirements of any company's migration regardless of the software application packaging methodology.

The migration process documentation will be implemented and incorporated in the system. A tracking system that has been specifically designed for application management and deployment will not only focus on the individual methods that have been seen in packages such as the Installshield Tracking System. It will also include an overview from a developer's perspective as well as a high level view so it can be

adapted to all Application Migration projects and integrated in any company at any time.

Aside from the development side of things, it is of the understanding that proper documentation will go hand in hand with the development work. There are several important documents required here.

- Requirements Documentation

- Database Structure and Process documents

- Test cases

- Coding documentation and structure

- End User Results and Implementation

- Conclusion and Modifications that could be made given time

A common problem in the package management area is that this area is very specialised. There are a number of different products such as DevTrack and Installshield's application management tool that have already been explained in Chapter 2. The basic concept is to develop a product that can be utilised in this area and maintained for further migrations.

Windows XP is the OS of choice for many organisations now but in a year's time Windows Vista is being deployed and applications are going to be updated and created for this system and there is an ever growing need for this specialised type of solution.

3.2   Project Life Cycle

We need to understand what Systems Development Life Cycle (SDLC) means. We
can give a good understanding by looking towards an excellent source of reference
below.

"The systems development life cycle (SDLC) is a conceptual model used in
project management that describes the stages involved in an information system
development project, from an initial feasibility study through maintenance of the
completed application. Various SDLC methodologies have been developed to
guide the processes involved, including the waterfall model (which was the
original SDLC method); rapid application development (RAD); joint application
development (JAD); the fountain model; the spiral model; build and fix; and
synchronize-and-stabilize.

Often, several models are combined into some sort of hybrid methodology.
Documentation is crucial regardless of the type of model chosen or devised for
any application, and is usually done in parallel with the development process.
Some methods work better for specific types of projects, but in the final analysis,
the most important factor for the success of a project may be how closely the
particular plan was followed."

[Quality Control and Software Process models. Software Engineering 8th Edition by
Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-
andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html]

There are a number of different stages in the SDLC methodology. We will now briefly review these stages. Deficiencies in an existing system are identified through interviewing users and consulting with key support personnel. New system requirements can then be defined. These may include addressing any of the deficiencies in the current system and giving specific proposals for improvements. When the proposed system has been designed, plans can then be created. These plans will detail the operating systems, programming, hardware and security issues.

The new components and programs must be acquired and installed. Users of the system must be trained to use the new product. All of the aspects of performance should be tested and adjustments must then be made, if deemed necessary. The system can then be put into use. There various methods to achieve this. We can phase in the new system: After a period of time, once the new system has been accepted and users are comfortable with using it, we can then gradually replace the old system and decommission it. Sometimes, it may be more cost-effective to completely remove the old system and implement the new system immediately.

Once the new system is in full use for a period of time, it should be evaluated. The system should be strictly maintained and users should be informed of any new modifications and procedures.

There are many different software processes one can use including, the Waterfall model, Evolutionary Development, Component-based software engineering. There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to a design that can be implemented.

Spiral Process Model

The Spiral process is the one chosen for this system. Its process is represented as a spiral rather then as a sequence of activities with backtracking reach loop in the spiral represents a phase in the process.

Reno fixed phases such as specification or design – loops in the spiral are chosen depending on what are required risks are explicitly assessed and resolved throughout the process.

A structured set of activities required to develop a software system are specification, design, validation and evolution. A software process model is an abstract representation of a process. It presents a description of a process from a particular perspective.

Process is represented as a spiral rather than as a sequence of activities with backtracking. Each loop in the spiral represents a phase in the process. There are no fixed phases such as specification or design. Loops in the spiral are chosen depending on what is required in each stage. Risks are explicitly assessed and resolved throughout the process cycle.

Determine objectives,
alternatives and
constraints

Evaluate alternatives,
identify, resolve risks

Risk
analysis

Risk
analysis

Risk
analysis

Opera-
tional
protoype

Prototype 3

Prototype 2

Risk
analysis

Proto-
type 1

REVIEW

Requirements plan
Life-cycle plan

Simulations, models, benchmarks

Concept of
Operation

S/W
requirements

Product
design

Detailed
design

Requirement
validation

Development
plan

Design
V&V

Code

Unit test

Integration
test

Integration
and test plan

Acceptance
test

Service

Develop, verify
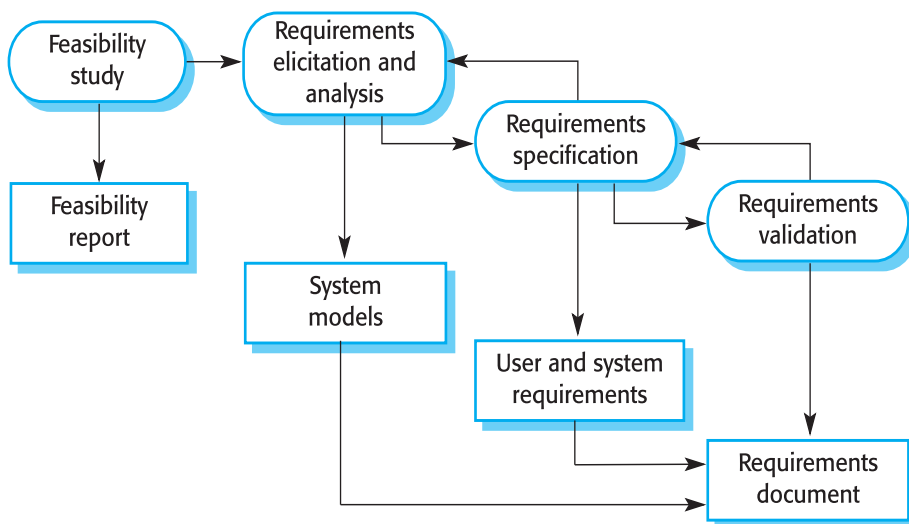next-level product

Plan next phase

Figure II Spiral Model

[Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html]

During the objective setting, specific objectives for the phase are identified.   We need to understand risk assessment and reduction where the risks are assessed and activities put in place to reduce the key risks. Moving on from there we can discuss the development and validation stages where a development model for the system is chosen that can be any of the generic models that we can briefly discuss.

Finally we have planning, the project is reviewed and the next phase of the spiral is planned. All of these stages can iterate until we have a successfully developed project.

There is a specific requirements engineering process that we will follow to gather requirements. The requirements engineering process is described in the below in Figure III.

Figure III Requirements Engineering Process

[Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html]

This is the requirement process we have chosen as the best solution to use to gather the requirements throughout the system. We can achieve the best results from following this type of model

3.3   Software Models

This section describes the software models we are going to use in our system design and implementation. UML (Unified Modeling Language) is a standard that can be employed to design a system properly.

"What is UML?

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software."

[What   is   UML?   Retrieved   on   the   08th   May   2007   from http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm]

There are many different goals with using UML for design purposes. UML provides users with a flexible visual modeling language that can be used to develop and exchange meaningful models. UML is an independent product that can be applied to

particular programming languages and development processes. We can gain a formal basis for understanding the UML modeling language.   UML also integrates best practices into its modeling techniques. It supports higher level development concepts. Finally it encourages the growth of object orientated tools. These are all important goals of UML. We will use of UML during the design phase.

Using UML we will develop Software models for the system.   There are many different levels of diagrams used in UML development. We are only going to discuss a few of these models that will be key to the design and implementation of this system. The first model we will consider is Data Model or ERD.

Data Model or (Entity Relation Diagram) ERD

> "An entity-relation diagram (ERD), also referred to as a data model, typically shows the name of each entity, its list of attributes and relationships with other entities. Primary keys can be identified for each entity and foreign keys used to express referential relationships between entities."

[Software   Models   Retrieved   on   the   17th   May   2007   from http://www.excelsoftware.com/models.html]

Often the data model is later implemented by generating SQL for an RDBMS (Relational Database Management System, such as Oracle) where entities become tables and attributes becomes columns. We will develop the ERD of this system in the next chapter.

Class Model

The class model shows static class objects in a system and the relationships between them. Two particularly important relationships are generalisation, this is also known as inheritance and aggregation or in other words the whole-part. Each class object on the diagram often shows the class name, its attributes and operations. The class model is usually a living model it will change thought out the modeling and development processes.

Details like data types for attributes and arguments for operations can also be shown on the diagram for some notations. These will be used to describe all of the Java code written. It is important to understand the use of the class model to correctly define the structure of the Class objects and implementations, so that the system can not only be designed properly but will be more efficiently developed.

Once we have designed the class models we can then develop use case models. We will now go on to discuss the use case model giving a brief description to understand the need to design them.

Use Case Model

Use case models are essential for developing an understanding on the users of this the system and how they use it, it gives a flow of control and functionality to the developers. Designing these models can be very beneficial. We will discuss this later in the implementation of the system.

"A Use Case Model describes the proposed functionality of a new system. A Use Case represents a discrete unit of interaction between a user (human or

machine) and the system. This interaction is a single unit of meaningful work, such as Create Account or View Account Details.

Each Use Case describes the functionality to be built in the proposed system, which can include another Use Case's functionality or extend another Use Case with its own behavior."

[The Use Case Model Retrieved on the 17th May 2007 from http://www.sparxsystems.com.au/resources/tutorial/use_case_model.html]
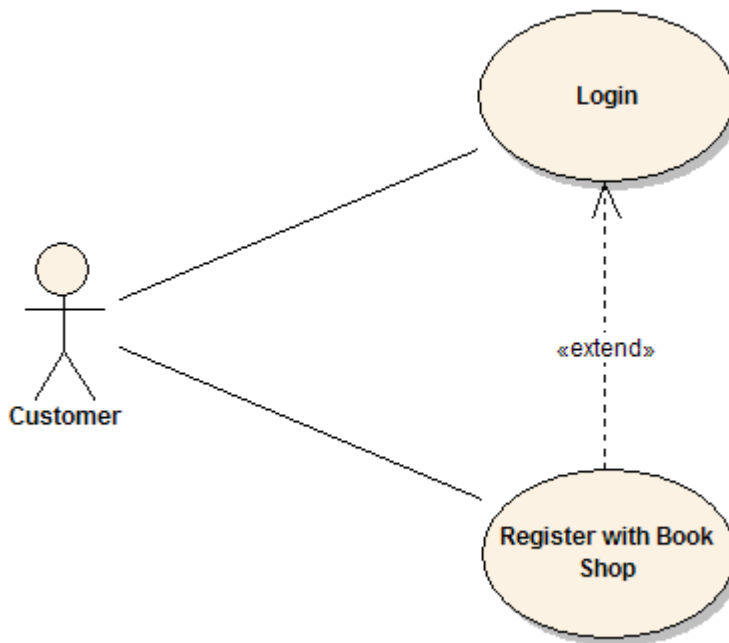
Figure IV Example of a Use Case

[The Use Case Model Retrieved on the 17th May 2007 from http://www.sparxsystems.com.au/resources/tutorial/use_case_model.html]

A Use Case description will generally include many different sections. Comments and notes describing the use case are almost essential. These are very useful to understand what the use case is about, and where they are needed.

Use Cases include a requirements description that are the formal functional requirements of the case that a use case must provide to the end user, such as <ability to delete a purchase>. These will then correspond to the functional specifications found in structured methodologies. That forms a constraint on the use case that must perform an action on the system when initiated.

Constraints can be described as formal rules and limitations that a use case operates under, defining what can and cannot be done in the system. These include pre-conditions that must have already occurred or be in place before the use case is run; for example, <create user> must precede <login>. We can also have post-conditions that must be true once the use case is completed. Finally we have invariants that must always be true throughout the time the use case operates.

Scenarios are another description that is also formal, sequential descriptions of the steps taken to carry out the use case. These can include multiple scenarios, to cater for exceptional circumstances and alternative processing paths.

3.4   Review of Deliverables

Requirements Documentation

The requirements document can actually be found in the introduction of this paper. These will be under consent review as the system may need to be revised due to user and developers conflicting views of the system.

The requirements are open to review and if deemed necessary they can and will be adopted to meet the required changes. We are following the Software Requirements process (please go to Figure III), to develop the requirements document so they are fully understood.

Database Structure and Process documents

These deliverables will be the actual design of the database for its referential integrity ("Referential integrity is a database concept that ensures that relationships between tables remain consistent.") and database table design and catalog. It is necessary to document what the data the database will be required store.   The process documentation is something that will go hand and hand with the design so that one can understand the reasons for developing it before developing it.

["Referential Integrity" - From Mike Chapple, Your Guide to Databases. Retrieved on the   20th   August   2007   from http://databases.about.com/cs/administration/g/refintegrity.htm]

Test Cases:

UML test case diagrams will be developed to allow us to understand what exactly the

system will do, and who will be able to do this. Software tools such as Rational Rose,

developed by IBM, ([IBM Rational Rose -Retrieved on the 20th August 2007 from

http://www-306.ibm.com/software/rational]) will be used to construct these diagrams

and they will be used to develop and understand what is needed for us to output to the

user. Test cases are often overlooked and deemed unnecessary, developers see it as

an extra unneeded documentation and diagrams that can be ignored.   However

without test cases the developers can get confused as to what the user requirements

are.

Coding documentation and structure

When using UML documentation we need to define the design of the classes we will

be using in Java. We have already discussed IBM Rational Rose tool, we will use to

develop these Class design diagrams. We can then develop an understanding on how

the code will be structured and how to proceed with development. When the class

diagrams have been created we can then begin developing the code and structure of

the application front-end GUI around this design. We can also include GUI diagrams

that can be used to prototype the capabilities of the final project.

End User Results and Implementation

The end user results and implementation, is merely UAT (User Acceptance Testing),

it will be documented by the user, the specific tests they have carried out and whether

or not it is good enough to implement into the production environment. It is at this

stage the user will give the acceptance to go ahead and sign off on the use of this system. They may also reject the system informing us of the issue found and improvements needed to be made in the system.

Conclusion and Modifications

This will state the conclusions of the overall project whether or not it met the user's expectation and whether it needs improvements and what these improvements are. Like all software there is always going to be a need for improvements and modifications.

3.5   Quality Control

Quality control is necessary to ensure that the system is being developed correctly. It is used for minimising the uncertainty and potential loss associated with the risk of errors. In other words it helps make sure that the system is running smoothly. Quality control is used in the software development process to help ensure that quality assurance procedures and standards are being followed. There are three types of quality reviews used:

1. Design or program inspections

2. Progress reviews

3. Quality reviews

Design or program inspections:

This is used to detect detailed errors in the requirements, design or code. A checklist of possible errors should drive the review.

Progress reviews:

The progress reviews are used to provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.

Quality reviews:

These are used to carry out a technical analysis of product components or documentation to find mismatches between the specification and the component

design, code to ensure that defined quality standards have been followed. Ways that we will be ensuring the quality of the system are the use of application and defect testing, and black box testing. When each section of the code has been completed the code will be put through a process that will try to obtain a crash on the program in any way possible. This will help highlight any errors that might not have otherwise been noticed. Debugging will also be used to check for errors in the code.

Black-box testing:

An approach to testing where the program is considered as a 'black-box' is one that the program test cases are based on the system specification Test planning can begin early in the software process.
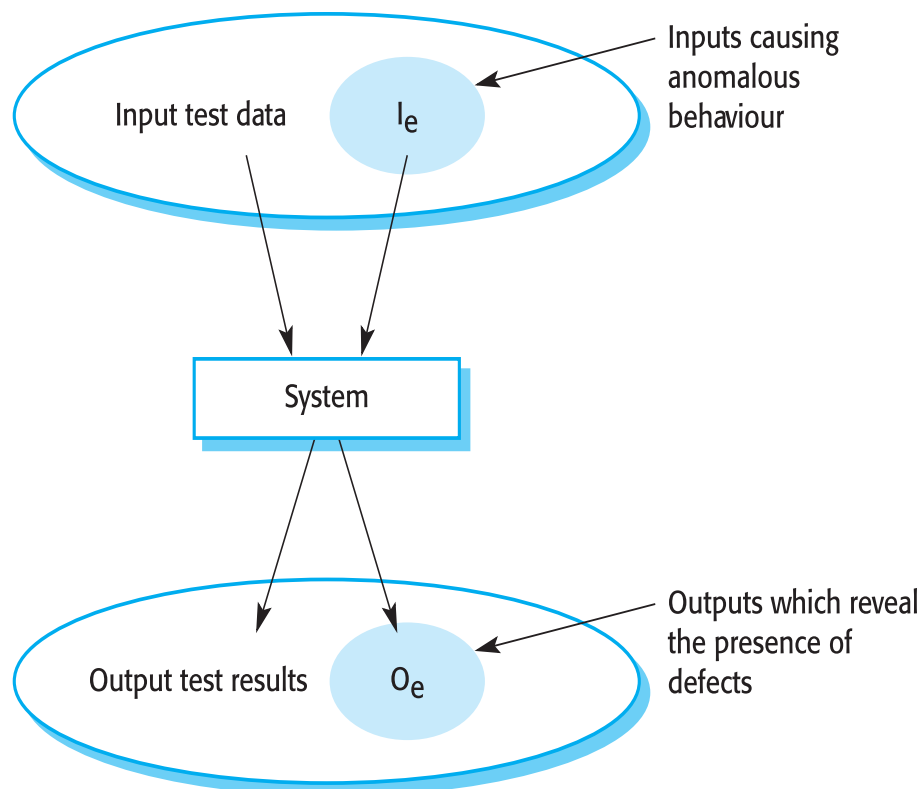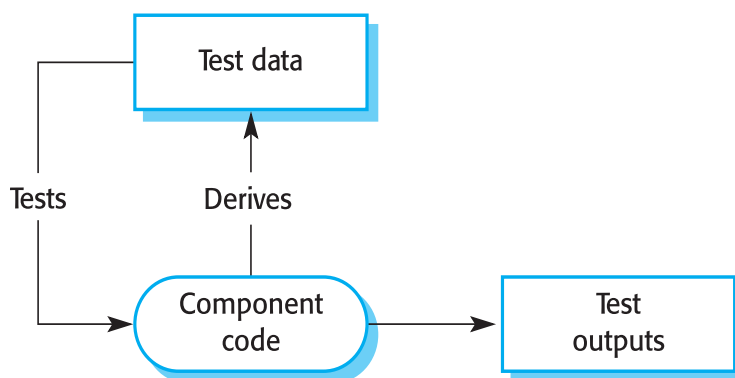

Figure V Black Box testing

[Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html]

Structural testing or white-box testing:

This is derivation of test cases according to program structure. Knowledge of the program is used to identify additional test cases needed. The objective is to exercise all program statements but not all path combinations should be tested.

Test data

Tests   Derives

Component code   Test outputs

Figure VI White Box Testing

[Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html]

Defect testing:

The goal of defect testing is to discover the defects in programs before they reach another process level. In saying that a successful defect test is a test that will cause a program to behave in an anomalous way Tests show the presence not the absence of these bug defects.

Testing priorities

Only exhaustive testing can show a program is free from bug defects. However, exhaustive testing is impossible. Tests should exercise a system's capabilities rather than its individual components. Testing old capabilities is much more important than the testing new capabilities that have been added. Testing typical situations is again   more   important   than the boundary value cases. Blackbox testing and white box testing are the two approaches that will be used in this project to ensure the application meets its requirements.

3.6   Goals/Outcome

The goals of this project are to document and to deliver an Application Packaging Tracking System. This is the primary task as the software deployment helpdesk departments are lacking a tracking inventory tool that will deliver a better understanding of the users in the system and the application required for these users and departments.

This project hopes to deliver a tool that can be implemented for any business environments software development and deployment process. It is a tool that is not currently commercially available. There are several different tool already developed on the same lines as this but none of the transportable they are too generic and lacking structure from one side or another, these have already been discussed in Chapter 2.

Once this project has been completed we should have a well documented and structured front end and backend system. This can be maintained and updated. In this chapter we have discussed five key areas. Deliverables of this project what this project will hope to deliver once completed, documents, the code, and the product. We then discussed the Project Life Cycle, the spiral model was selected for use in the development of this system, we then went on to discuss the requirements model that would be used for gathering and maintaining the requirements of the system.

In Section 3.3, we discussed the software design models that we would need to create in order to design and develop this project. It was decided that UML would be the modeling language approach. We discussed various important models in this section. Following on from this, we introduced the review of the current deliverables. Finally,

we took a briefly reviewed the quality control methods that would be adopted in order to successfully complete and continually review the product quality of this product.

4   Design and Implementation

4.1   Project Background

In Chapter 3, we have investigated the methodologies behind application packaging tracking system, what we will use to design and test this system. We had concluded in Chapter 2 that the application packaging tracking system is a toolset that is missing from the application packaging and deployment process. In this chapter, we are going to discuss how we will design and develop this tool.

There are existing applications in this area. However, no existing solution is available that is customised to meet an overall solution with all the generic requirements that this system has proposed. We have discussed the various available solutions, some very good but none that cover all of the process area. Most current systems deal with either tracking the lifecycle of the application or tracking a particular method of deploying or creating the application. The specific requirement that moulds all these together in a more simplified but effective tool does not exist.

Through the process of designing and developing this system we will achieve and meet the goals of this required generic system. In this chapter, we will examine the project schedule and milestones. We will also develop the database design and structure that will be shown in this chapter to understand the systems data store.

Once the backend database structure has been designed and implemented, the design of the Graphical User Interface (GUI) becomes important. We will also discuss the Java GUI implementation in this chapter, and finally finishing with a GUI Prototype the system.

4.2 Project Management and Schedule

ID		Task Name	Duration	Start	Finish	Predecessors	'07 W T F S S	19 Feb '07 M T W T F S S
1		**Research**	**37 days**	**Thu 21/09/06**	**Fri 10/11/06**			
2		Review Market Leading Products	10 days	Thu 21/09/06	Wed 04/10/06			
3		User Requirements	5 days	Thu 05/10/06	Wed 11/10/06	2		
4		Back End Research	10 days	Thu 12/10/06	Wed 25/10/06	3		
5		Front End Research	10 days	Thu 26/10/06	Wed 08/11/06	4		
6		System Requirements	2 days	Thu 09/11/06	Fri 10/11/06	5		
7		Produce Requirements Document	2 days	Thu 09/11/06	Fri 10/11/06	5		
8		Research Completion	0 days	Fri 10/11/06	Fri 10/11/06			
9		**Development**	**122 days**	**Fri 10/11/06**	**Mon 30/04/07**	8		
10		DB Backend Development	18 days	Fri 10/11/06	Tue 05/12/06			
11		Front End Coding	54 days	Wed 06/12/06	Mon 19/02/07	10		
12		Testing & Tweaking	18 days	Tue 20/02/07	Thu 15/03/07	11		
13		Final Development QA	32 days	Fri 16/03/07	Mon 30/04/07	12		
14		Development Completion	0 days	Mon 30/04/07	Mon 30/04/07			
15		**Documentation**	**181 days**	**Fri 10/11/06**	**Fri 20/07/07**	8		
16		DB Processes & Structure	18 days	Fri 10/11/06	Tue 05/12/06	8		
17		Coding Documentation	20 days	Thu 01/02/07	Wed 28/02/07			
18		Outcomes/Research Tracking System results and Implen	35 days	Thu 01/03/07	Wed 18/04/07	17		
19		Diagrams/ processes	35 days	Thu 19/04/07	Wed 06/06/07	18		
20		Explanations/ Structuring	32 days	Thu 07/06/07	Fri 20/07/07	19		
21		Documentation Completion	0 days	Fri 20/07/07	Fri 20/07/07			
22		**Documentation Review**	**25 days**	**Mon 23/07/07**	**Fri 24/08/07**	21		
23		Conclusions & Complete Documentation	25 days	Mon 23/07/07	Fri 24/08/07			
24		Project Completion	0 days	Fri 24/08/07	Fri 24/08/07			

Figure VII Project Management and Schedule

Figure VII shows the project management and schedule developed to enable us to structure our outcomes and development tasks.   It is also important that documentation is created concurrently with development.

4.3   Milestones

As we can see from the project plan in Section 4.2 we have four main milestones. These include research, development, documentation and documentation review. These four areas have been defined as the main milestone areas. Each milestone is a critical goal that must be achieved so that we can design and develop this system.

However, we can further break it down and explain each section. Under the research milestone there are two main areas we needed to research, the backend (Oracle Database) and coding language and technologies used in (Java). These would be the research milestones to finish and understand what we need to do to develop this project. The development milestone again we have two main areas the Oracle Database design and implementation and the GUI development using Java.

Documentation is an ongoing process. Documents will always be considered to be live documents even after the project has been completed. The primarily reason of this is that people will need to review, update or maintain the system and must understand how and why it was developed.

The final milestone is documentation review, this involves a high level review of the documentation written and any processes or section that needs to be updated will be done at this stage. Documentation review is a very important milestone as we need to

understand what changes have been made and update the documents to reflect these changes. Once this has been completed they can be published for acceptance and approval sign off.

4.4 Database Design

The database structure is the basic most important design that is required. We have already agreed that it will be developed in Oracle SQL language. What is now required is the development of the structure and content of the database. In particular, we need to develop a design of the tables and relationship with a database catalog and an Entity Relationship Diagram (ERD).

The remainder of this section discusses the database structure. We will first discover the table structure, what tables need to be design and what data needs store. We will develop a catalog of this database, and then we will design and create an Entity Relationship Diagram. This will help define how the database will link together in order to fit the requirements

Once the relationalships and the database structures have been developed, the next step is to implement this design in SQL code. The SQL code can be found in Appendix A. This will result in a database structure that exists within the Oracle environment. The next step will be to develop the GUI. This will be discussed in more detail in Section 4.6.

4.4.1   Data Normalization:

As we have already discussed in Section 4.3, we need to understand what is required from the system to do this correctly we need to follow what is known as normalization. Here is a brief look at what normalization is and how we can apply it.

"There are three main normal forms, each with increasing levels of normalization:

- First Normal Form (1NF): Each field in a table contains different information. For example, in an employee list, each table would contain only one birthdate field.

- Second Normal Form (2NF): Each field in a table that is not a determiner of the contents of another field must itself be a function of the other fields in the table.

- Third Normal Form (3NF): No duplicate information is permitted. So, for example, if two tables both require a birthdate field, the birthdate information would be separated into a separate table, and the two other tables would then access the birthdate information via an index field in the birthdate table. Any change to a birthdate would automatically be reflect in all tables that link to the birthdate table."

[Normalization Retrieved on the 19th May 2007 from http://www.webopedia.com/TERM/N/normalization.html]

It is not in scope of this project to investigate normalization in greater detail, it is more important to understand the conclusions were drawn and, therefore, the implementation   of   the   Database   shown   below.

4.4.2 Database Catalog

Table I shows the representation of the database structure, these are all the tables created that are need for the system to operate to the business and user requirements. As can be readily seen there is a break down of each field and attributes required for each individual table, and the individual fields that are needed.

Table Names & Attributes	Description	Field Attributes	Required/Optional
App_TBL	**Application Table**		
App_ID	Application ID - Stores the Primary Key	Number(10)	Yes
Vendor_ID	Stores the Vendor ID and is a Foreign Key to Vendor Table	Number(10)	Yes
Language_ID	Foreign Key to Link to Language Table	Number(10)	Yes
App_Name	Application Name	VARCHAR2(20)	Yes
App_Ver	Application Version	Number(10)	Yes
Rev_No.	Revision Number	Number(10)	Yes
Type_ID	Whether it's a Softricity Application Or MSI/SMS	Number(10)	Yes
License_ID	Links License Table Foregin Key	Number(10)	Optional
Vendor_TBL	**Vendor Table**		
Vendor_ID	Primary Key	Number(10)	Yes
Vendor_Name	Vendor Name	VARCHAR2(10)	Yes
Contact_Name	Vendor Contact Name	VARCHAR2(20)	Yes
Email_Add	Email Address If Given	VARCHAR2(30)	Optional
Vendor_URL	Vendor Website address for support	VARCHAR2(60)	Optional
Phone_No	Vendor Support Phone No	Number(10)	Optional
AppType_TBL	**Application Type**		

| Type_ID | Primary Key | Number(10) | Yes |
| Type | Softricity/MSI | VARCHAR2(10) | Yes
Yes(Default
Softricity) |

Dev_TBL	**Developers Table**		
Dev_ID	Primary Key	Number(10)	Yes
Fname	FirstName	VARCHAR2(20)	Yes
Lname	LastName	VARCHAR2(20)	Yes
DevType_ID	Developer Type_ ID - Foreign Key	Number(10)	Yes

DevType_TBL	**Developer Type Table**		
DevType_ID	Developer Type_ ID Primary Key	Number(10)	Yes
Dtype	Developer Type - Tester/Developer/Discovery/Deployment	VARCHAR2(20)	Yes

Dept_TBL	**Department Table**		
Dept_ID	Primary Key	Number(10)	Yes
Dept_Type	Department Name	VARCHAR2(20)	Yes

Status_TBL	**Application Status Table**		
Status_ID	Primary Key	Number(10)	Yes
Status_Type	Delayed/UAT/Deployed/QA/Deveploment/Discovery	VARCHAR2(20)	Yes

Users_TBL	**User Table**		
User_ID	User ID Primary Key	Number(10)	Yes
UserS_ID	UserStatus ID Linked Foreign Key	Number(10)	Yes
Fname	Firstname	VARCHAR2(20)	Yes
Lname	Lastname	VARCHAR2(20)	Yes
Dept_ID	Department ID Foreign Key	Number(10)	Yes
User_Acc	User AD Account Name ie Cquillinan	VARCHAR2(20)	Optional
Email_Add	User Email Address	VARCHAR2(40)	Yes
Phone_No	Phone Number	Number(10)	Optional

Location	Location LONDON, PARIS DUBLIN	VARCHAR2(20)	Yes
Floor	1st 2nd etc	Number(10)	Optional
Manager_ID	this links to User table as manager will be a User	Number(10)	Optional
Job_Title	Job Title Not really required	VARCHAR2(20)	Optional
UserStat_TBL	**User Status Table**		
UserS_ID	Primary Key	Number(10)	Yes
Status	Whether there deployed or not to UniBuild	VARCHAR(10)	Yes
Lang_TBL	**Language Table**		
Language_ID	Primary Key	Number(10)	Yes
Language_Code	ENG for English, GER for GERMAN	VARCHAR2(5)	Yes
Language_Name	Description as GERMAN for CODE GER etc	VARCHAR2(20)	Yes
Lic_TBL	**Application License Table**		
License_ID	License ID Primary Key	Number(10)	Yes
License_Type	Type Corporate standalone	VARCHAR2(20)	Yes
License_Code	License ID	VARCHAR2(20)	Yes
License_Quant	Quantity allowed to Use	Number(10)	Yes
ADGroup TBL	**AD Group Table**		
ADGroup_ID	AD Group Primary Key	Number(10)	Yes
ADGroup_Name	AD Group Name	VARCHAR2(64)	Yes
Shortcut_Name	Shortcut it references	VARCHAR2(20)	Yes
App_ID	Application ID Foregin Key	Number(10)	Yes
Priority_TBL	**Priority Table**		
Priority_ID	Primary Key	Number(10)	Yes
Priority_Name	High/Medium/LOW	VARCHAR2(20)	
AppPro_TBL	**Application Process Table**		

AppPro_ID	Primary Key	Number(10)	Yes
App_ID	Foreign Key	Number(10)	Yes
Dev_ID	Foreign Key	Number(10)	Yes
Status_ID	Foreign Key	Number(10)	Yes
Tester_ID	Links the Dev_ID also	Number(10)	Yes
Priority_ID	Foregin Key	Number(10)	Yes
Request_Date	Date it was requested	Date	Yes
Test_Date	Date Tested	Date	Yes
Deploy_Date	Deployed Date	Date	Yes
Notes	Any comments or Notes	VARCHAR(60)	Optional
Disc_TBL	**Discovery Table**		
Disc_ID	Primary Key	Number(10)	Yes
App_ID	Application link Foreign Key	Number(10)	Yes
Dev_ID	Developer ID who Discovered it	Number(10)	Yes
Doc_Link	Documentation Link URL	VARCHAR(80)	Yes
DeptApp_TBL	**Required Apps in Department**		
DeptA_ID	Primary Key	Number(10)	Yes
Dept_ID	Department ID Foreign Key	Number(10)	Yes
App_ID		Number(10)	Yes
UserDApp_TBL	**User Required Applications**		
USerDApp_ID	Primary Key	Number(10)	Yes
App_ID	Application Link Foreign Key	Number(10)	Yes
User_ID	User Table Foreign Key	Number(10)	Yes

Table I Application Packaging Database Catalog

4.4.3   Entity Relationship Diagram (ERD)

Figure VIII shows the physical view of the relationships between the tables and there

linkages. As we can see there is no one table alone in the system. However, there is

room for a login users table. This has currently out of scope of this project due to

time constraints. This is an optional extra that we will discuss and implement at a
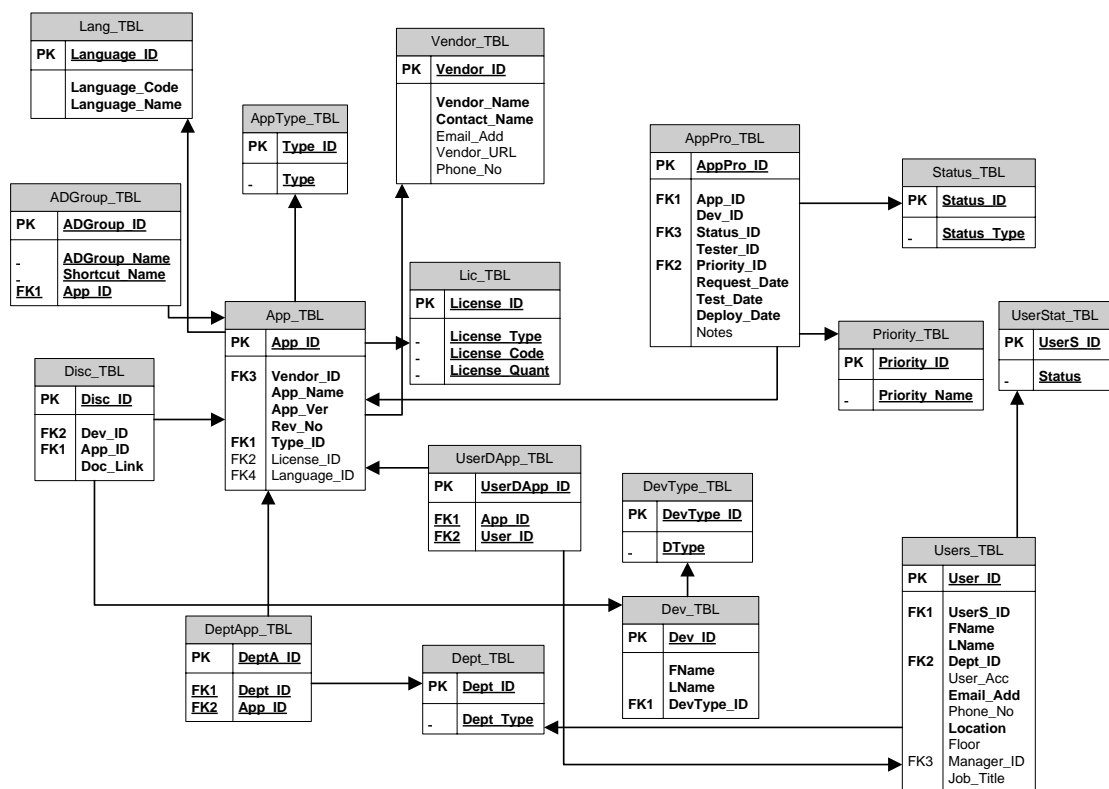
later date.

**Figure VIII Entity Relationship Diagram (ERD) of Application Tracking
System.**

4.5   Software Implementation

When we discussed how we where going to go about developing this project we understood there was going to be two major components. One is the data storage. This has been dealt with by the Oracle SQL database in Section 4.4. As we now have developed the database backend, we will now proceed with the GUI front-end.

In order to achieve this, we must first understand the requirements for the GUI front-end. We need to analyse what classes are required to be develop and what structure should be follow. We already know where, and how, the data will be stored. We now need to know how to extract that data in a meaningful manner. We will also need to be able to update, delete, save the data. There needs to be a flow that will make the user's daily life easier.

4.5.1   Code Design

Code design is a broad topic area. There are so many different models in this area.

However, in this dissertation we are primarily interested in UML ([9] what is UML?])

use case and class diagrams.   This is how we structure object orientated (OO)

languages, such as Java.

UML Complete Use Case diagram
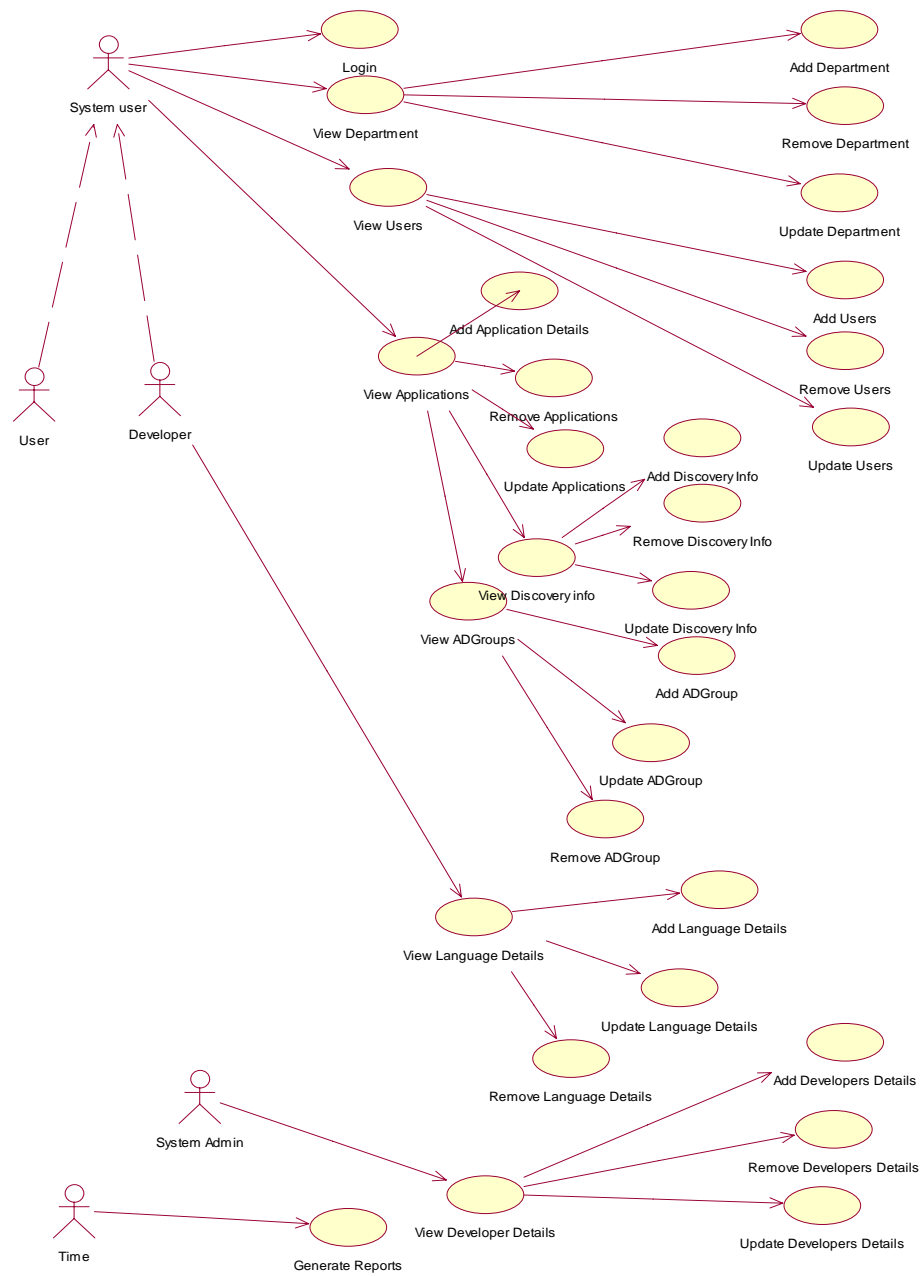
Figure IX Application Packaging Tracking System Use Case

Figure IX displays the desired flow of the system. However, we will explore each flow of control individually. We will give a representative number of examples below. The entire Use Case analysis is available in Appendix B.

Use Case – User Login

Figure X shows us how a system user would login to the system. We can see from below there are two types of system users, a User and a Developer. The pre-condition is that the user must already be apart of the system.

Login

System user
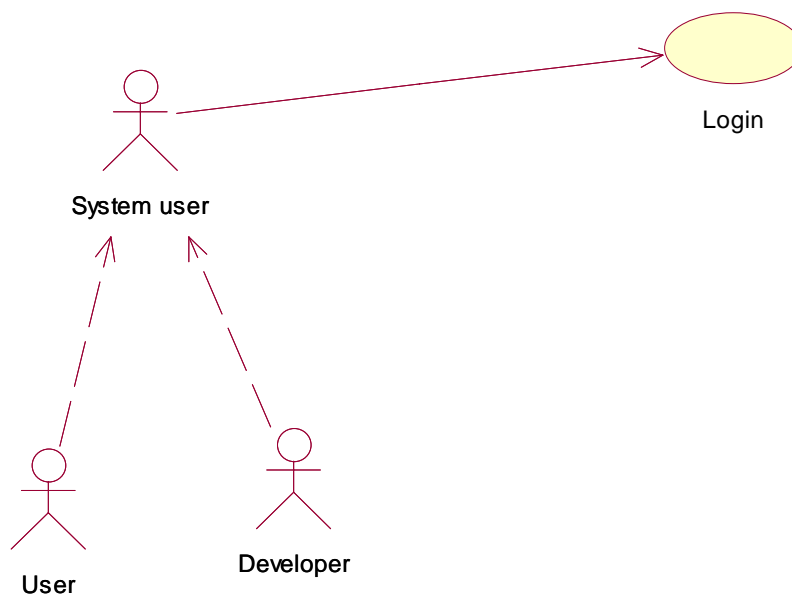
User

Developer

Figure X User Login Use Case

Use Case	Detail
Name	Login
ID	1.0
Description	Allows a user to login to the system
Primary Actors	System User (User and Developer)
Secondary Actors	None
Preconditions	User must be in the system
Main Flow	1. User enters their details into the system and login
Post Conditions	Login accepted or denied
Alternative Flow	None

Use Case – View Applications and View ADGroups
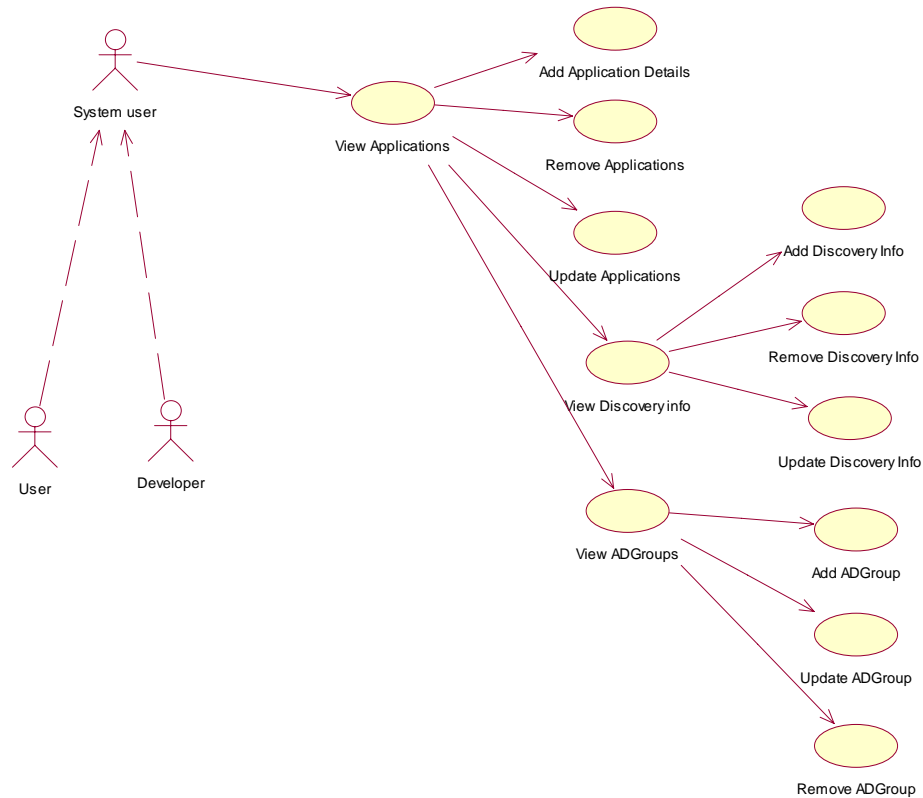
Figure XI View Applications and View ADGroups Use Case

Figure XI describes how a user can view the applications in the system and what actions they can perform. We can see two of the actions are to *View ADGroups* and *ViewDiscoveryInfo*. These actions can take a different path that has other actions underneath them. Below are the Use Case explanations detailing the above Figure XI.

Use Case	Detail
Name	View Applications
ID	4.0
Description	Allows a user of this system to view all the Applications in the company's system
Primary Actors	System user (User and Developer)

Secondary Actors	None
Preconditions	User must be login successfully

Main Flow	1. Details can be displayed on screen for Applications information.
	
	1. User can browse this information
	
	2. Users can view the application information.
	
	3. Developers can
	
	    a. Add Application details
	
	    b. Remove Application details
	
	    c. Update Application details

Post Conditions	Application information displayed on screen. Developers can proceed with add/remove and update functions only
Alternative Flow	Add Application Details Remove Application Details Update Application Details View Discovery Info View ADGroups

Use Case	Detail
Name	Add Application Details
ID	4.0.1.1
Description	Allows a user of this system to add new Application information

Primary Actors	Developer ONLY
Secondary Actors	None
Preconditions	User must be login successfully Must be in the Application view page
Main Flow	1. New Application information can be added into the system
Post Conditions	New Application information added successfully
Alternative Flow	None

Use Case	Detail
Name	Remove Application Details
ID	4.0.1.2
Description	Allows a user of this system to remove a Application information
Primary Actors	Developer ONLY
Secondary Actors	None
Preconditions	User must be login successfully Must be in the Application view page
Main Flow	1. Delete Application Info on the system
Post Conditions	Application information is removed from the system
Alternative Flow	None

Use Case	Detail
Name	Update Application Details

ID	4.0.1.3
Description	Allows a user of this system to Update particular Application information.
Primary Actors	Developer Only
Secondary Actors	None
Preconditions	User must be login successfully Must be in the Application view page
Main Flow	1. Update (Save) Application information
Post Conditions	Application Info is updated
Alternative Flow	None

Use Case	Detail
Name	View Discovery Info
ID	4.0.2
Description	Allows a user of this system to view the Discovery information for this application
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully
Main Flow	1. Details can be displayed on screen for the discovery information for this particular application. 2. User can browse this information 3. New Discovery information can be added into the system

	4. Information can be removed from the system
	5. The data can be updated(saved)
Post Conditions	Discovery information displayed on screen
Alternative Flow	Add Discovery Info
	Remove Discovery Info
	Update Discovery Info

Use Case	Detail
Name	Add Discovery Info
ID	4.0.2.1
Description	Allows a user of this system to add new Discovery information
Primary Actors	System User (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully
	Must be in the Discovery view page
Main Flow	1. New Discovery information can be added into the system
Post Conditions	New Discovery information added successfully
Alternative Flow	None

Use Case	Detail
Name	Remove Discovery Info
ID	4.0.2.2

Description	Allows a user of this system to remove a Discovery information
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully Must be in the Discovery view page
Main Flow	1. Delete Discovery Info on the system
Post Conditions	Discovery information is removed from the system
Alternative Flow	None

Use Case	Detail
Name	Update Discovery
ID	4.0.2.3
Description	Allows a user of this system to update particular Discovery information.
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully Must be in the Discovery view page
Main Flow	1. Update (Save) Discovery information
Post Conditions	Discovery Info is updated
Alternative Flow	None

Use Case	Detail
Name	View ADGroups
ID	4.0.3
Description	Allows a user of this system to view the ADGroups information for this application
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully, and the application in question must be displayed
Main Flow	1. Details can be displayed on screen for the ADGroup for that application. 2. User can browse this information 3. New ADGroup information can be added into the system 4. Information can be removed from the system 5. The data can be updated(saved)
Post Conditions	ADGroup information displayed on screen
Alternative Flow	Add ADGroup Remove ADGroup Update ADGroup

Use Case	Detail
Name	Add ADGroup
ID	4.0.3.1

Description	Allows a user of this system to add new ADGroup information
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully Must be in the ADGroup view page
Main Flow	1. New ADGroup information can be added into the system
Post Conditions	New ADGroup information added successfully
Alternative Flow	None

Use Case	Detail
Name	Remove ADGroup
ID	4.0.3.2
Description	Allows a user of this system to remove a ADGroup information
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully Must be in the ADGroup view page
Main Flow	1. Delete ADGroup Info on the system
Post Conditions	ADGroup information is removed from the system
Alternative Flow	None

Use Case	Detail
Name	Update ADGroup
ID	4.0.3.3
Description	Allows a user of this system to Update particular ADGroups information.
Primary Actors	System user (User and Developer)
Secondary Actors	None
Preconditions	User must be login successfully Must be in the ADGroup view page
Main Flow	1. Update (Save) ADGroup information
Post Conditions	ADGroup Info is updated
Alternative Flow	None

4.5.2   UML Class Diagrams

In this section, we are going to give a briefly outline into Class diagrams. Class diagrams are used to design classes that with will be developed within a programming language. We have selected Java to develop this system. Class diagrams describe the design of Java classes. We can then go from design to implementing with a complete structure of a class.

> "UML Class diagrams are the mainstay of object-oriented analysis and design. UML 2 class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling. Although I prefer to create class diagrams on whiteboards because simple tools are more inclusive most of the diagrams that I'll show in this article are drawn using a software-based drawing tool so you may see the exact notation."

[UML   2   Class   Diagrams   Retrieved   on   20th   May   2007   from http://www.agilemodeling.com/artifacts/classDiagram.htm]

We will now proceed to give a number of examples of class diagrams. These will be used during the development of this system. The first class is the Vendor class.
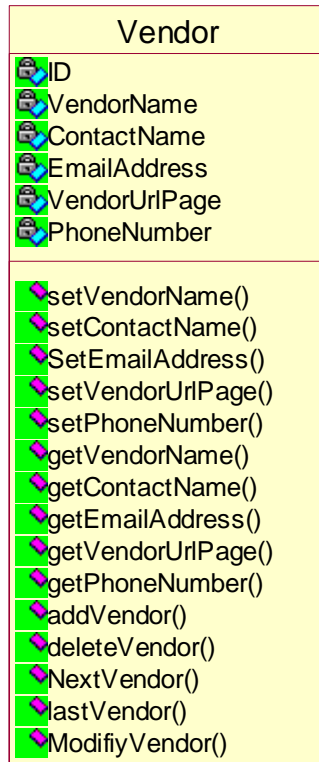
Figure XII Vendor Class

Figure XII shows the Vendor Class. This is a Class diagram representing a Vendor.

It has attributes at the top section with the methods that operate on these attributes

below. This design diagram will be implemented when creating the Java code.
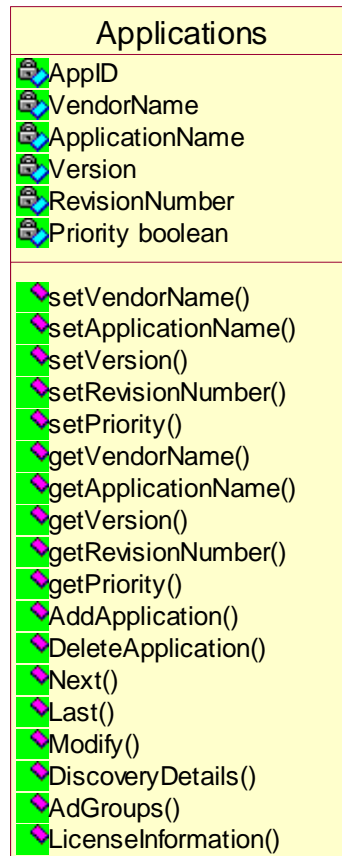
Figure XIII Application Class

Figure XIII shows Application class. As with Figure XII, it has attributes at the top with operations below.   We can see it has different functions such as *DiscoveryDetails*. This is an operation that will invoke the discovery screen.

We have only shown two class diagrams. The system requires a full set of class diagrams in order to complete the implementation.

4.6   GUI Design

This section describes the basic layout of the GUI. These are the prototypes used to understand how the system will look. Figure XIV shows the Login Dialog. This is the initial interface that a user will encounter.

Figure XIV Login Dialog Interface
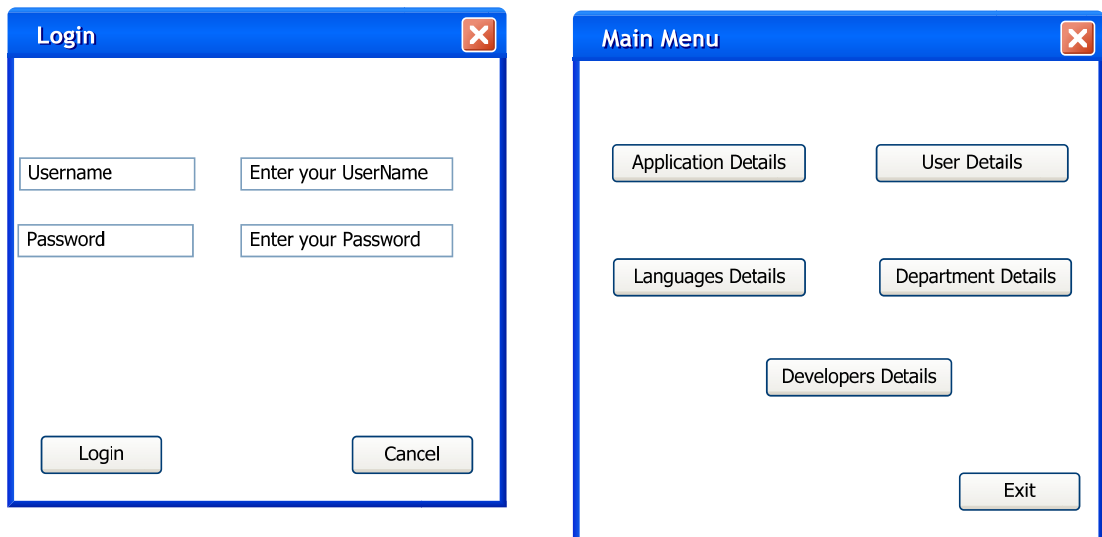
Figure XV Main Menu Interface

Figure XV shows the main menu giving us six different flows of control to use.

We will now describe the Application Details interface. The remaining GUI elements are described in Appendix C. The Application Details interface is shown in Figure XVI.   Using this interface we can browse to the Discovery Information and ADGroups designed for a specific application. There are also standard flow patterns first, last, previous and next, with the options to add and remove. The exit button should return the flow back to the main menu.

Figure XVI Application Details Interface

4.7   Chapter Summary

In this chapter, we have discussed many different elements of the design and implementation. We began by discussing the project schedule, setting out milestones and deliverables this project. The Section 4.3 develops the milestones, giving a briefly summarising them.

Section 4.4 began discussing the database design. In this section we develop an understanding about how to design the system with data normalization. The database catalog was then designed. This has been included in subsection 4.4.2. The ERD or, entity relationship diagram, gives a clearer understanding about the relationships between the tables in the data catalog.

Once the database design had been completed and implemented we proceeded to discussing the software implementation. We discussed the coding design using UML Use Case diagrams and Class diagrams, giving examples of each. Finally we discussed the GUI design using a prototype of the GUI.

5   Findings and Analysis

In the proceeding chapters, we have explored the process from the problem statement through to the methodology and implementation. We will now focus on the results obtained using this solution. We need to consider whether or not the proposed system meets the requirements of the application packaging and deployment process. We will also discuss issues that occurred and how we can manage these issues in the future. These are all questions that must be evaluated at this stage.

5.1   Analysis of Results

We set out to develop a proof of concept. This proof of concept is intended to solve the problem of tracking the application packaging and deployment process. The Application Packaging Tracking System is, in essence, a required business software inventory tool. The success of the IT infrastructure in any big business environment depends on either this type of toolset, or, at least, a systematic approach to infrastructure management.

We have investigated the problem area of a managed, tracked application packaging and deployment process in Chapter 2, and detailed the approach to design in Chapter 3. The Database structure shown in the Chapter 4 details the complexity of the system.

We have discovered from the development and implementation that it is insufficient just to provide a system that tracks the applications. Real-time data is required and a rigorous data management system must be used. Other non-functional concerns, such

as security, have not yet been thoroughly investigated. Further research and development is required in order to fully realise the potential of this project.

5.2   Evaluation

We have taken a scenario approach throughout this dissertation. We used the same scenario as a test case during development. In Chapter 2, we discovered that no existing products were available to completely manage the whole process area. Considering the idea of managing and correctly tracking this process was not a simple approach, but a required one.

The application was implemented with both a backend database and front-end GUI. The premise of this thesis is if you want to have a successful managed application deployment plan, then you need an integrated managed Application Packaging Tracking System. This system interprets the application packaging and deployment process and allows us to store and efficiently manage important data.

The application packaging and deployment process is now fully integrated with this system. The database structure discussed in Chapter 4 efficiently manages and stores all the data needed for this system. We also discussed in Chapter 4 the functionality aspects, with the front-end GUI design in mind. We have designed and developed a considerable amount of the functional aspects of the system needed to integrate this system into this process area. However, even though we have designed and developed this system, it has not been fully completed. There are a lot of potential improvements that can be applied the system.

5.3   Project Issues/Limitations

During the research and implementation phase, some limitations became apparent. While significant work was preformed evaluating the requirements of the Application Packaging Tracking System, clearly considering every scenario would be impossible. Some of these unforeseen limitations impacted the project. However, it did not lead to failure in delivering this proof of concept.

Gathering requirements from users is inherently difficult, as it is difficult to gain an understanding of every users needs. Time is always a basic limitation. Other limitations that were incurred were the lack of experienced resources in the development area. This delayed the development, but did not halt it.

System security was limited due to the lack of time and resources. Implementing a secure platform was one of the goals of this project. However, we do not have user entry security at this time. This will follow in the succeeding months. Naturally, more testing will help to improve the stability and features of the system. As this project was conceived as a proof of concept, it was considered more important to establish a working prototype rather then a polished product.

5.4   Project Benefits

In Section 5.3, we discussed the limitations of the current implementation of the project. In this section, we will, in contrast, discuss the benefits of the Application Packaging Tracking System. When we first looked at this problem area, it was immediately apparent that a solution in this area was required. Given this premise, the main benefit is that there is now a completely automated application packaging and deployment process and key data is tracked in the system.

This solution is unlike the modular solutions available in the commercial of-the-shelf world, such as those described in Chapter 2. These solutions include SMS, Devtrack and Installshield solutions. Such solutions are in more limited areas of the process and cannot been integrated with each other. This alone is a key benefit to this system, that it covers all of the process areas required for a successful application packaging and deployment system.

This project can be generalised from the current environment to others as a broader solution. It can be seen as a key solution to successful application management and deployment. We can also say that it is cross OS compatible as it has been developed with two specific technologies in mind, Java and SQL, that are not OS dependent.

5.5   Impacts on Project

When planning any project or development, there are always going to be extra tasks or unforeseen shifts in the timeline. These are usually due to issues and risks that arise during different phases of the project life cycle. This project is no different from any other development project. We have seen some issues and risks that have impacted on this system in Section 5.3. The complexity of the project grew throughout the research and development. This was unavoidable: understanding user and system requirements were the main reason for this growing complexity.

It was understood that not all of the front-end GUI would be developed in the project life cycle as this was only meant as a proof of concept. However, due to the development of the backend becoming more complex, the development of the front-end GUI became more complex also and more time consuming. This shifted the development timelines and meant that less was achieved in the time given. This did cause concern and extra work. However, it did not cause a huge amount of overall risk as we were only delivering a prototype solution.

The budgetary aspect was certainly an impact on the project solution. If a budget was given to this project, more experience resources could be dedicated towards the design and implementation of this tracking system and we could improve on the design and in turn this would have led to a better developed system. Given that resources were few and time was short, the initial idea behind this project was to prove that it will give any business or corporation with an application packaging and deployment process. This process will be more efficient and better constructed solution towards tracking the application packaging and deployment process.

5.6   Findings/Analysis – Outcome

This solution described in this thesis shows considerable advantages compared to the current application packaging and deployment process.   While throughout the research and development of this project the complexity grew, this has enhanced the system, but has also increased the value of the system.

When we began the research into the different areas, a broader prospective of the entire architecture was required. However, as the different sections were analysed we can see how each one links to each other. The current database development successfully represents the structure and data required for this system. This was necessary to provide for an efficient system.

From the security perspective, we are currently missing a user login authentication system. Concepts have been considered in this area. For example, using OS account authentication. Another possible approach is a separate security solution. Work on this area is ongoing.

The implementation work that has been completed on the system to date has shown that this is an invaluable system that is currently not available commercially. As stated in Chapter 1, 67% companies do not track their software. This solution allows a reduction in management overhead while increasing the understanding of the needs of departments and users. Overall the system has proven to be successful and integrated as a well needed managed process flow.

6   Conclusion

In Chapter 4, we examined the design and implementation of the Application Packaging Tracking System. We then examined findings and analysis in Chapter 5. This chapter will focus on the lessons learned from this experience. We will analyse the problems that occurred, discuss the results, and finally suggest future directions.

6.1   What we have learned

We must first address the lessons learned through the research and development of this project. The main contribution of this thesis is the development of an integrated solution to application package management. This solution provides a complete application packaging and deployment process management. Invaluable experience was gained from both a user and management perspective. Experience was also gained on how to develop an efficient managed system.

From a research and design prospective, understanding the information gathered was an ongoing process. The design was modified to reflect the experience gained during the research and implementation phases in order to reflect the needs of the users of the system. There was an extensive learning curve at the implementation stage. Challenges included the design and development of the database work. In particular, designing the tables to be generic but, yet fulfil the system requirements. Considerable knowledge has been gained in Java coding and development areas. New and different methods of development have been utilised. Understanding was gained in new methods and different technologies used in the continuing development of this system.

In general, the experience, from the problem definition to implementation solution, has been an interesting learning experience. We also discovered that there are many different approaches that could be taken. The optimum approach is often not immediately obvious. There is still more that can be learned from this solution, and it poses some interesting questions for the future.

6.2   What we do differently

It is always easier to review decisions with hindsight. There are many areas that a fuller understanding, gained through trial and error, would now lead to a different approach.

Without time limitations, it would have been preferable that, more research in this area would have been performed. Furthermore, the provision of a model for the business would have been preferable. Marketing is always an important function. Without the attractiveness of a product, it is difficult to expect to promote a theoretical idea.

Research could also have been more thorough. More research would have improved the implementation of the system. The solution described in this dissertation is by no means perfect. However, it has proven to be successful. The original intention, to provide a proof of concept, has been achieved. While there are simpler GUI based approaches that could have been used to develop the project faster. Given that this is going to be used primarily on windows based systems, we could have selected a different development language. This could have reduced the development time. However, it would have removed the advantage that this system is platform independent.

6.3   Summary of the Project Development and Documentation

The development began as expected in November 2006. The initial step was to research the relevant technologies.   Documentation was the backbone of the implementation. If you do not plan you will not develop an accurate and efficient system.

This background research was outlined in Chapter 2. We then had an understanding of the process area and could begin on the methodology. It needed to be decided what development method the system would use. The design and implementation phase followed.

The documentation was written at the same time as the design and implementation phase. The backend database structure was designed and completed using the SQL language; the design detail was documented with the database catalog and the ERD. This can be found in Chapter 4. The database design and code had to be revised until the structure met the user requirements.

Once the backend design and implementation code was written, it was then a task of implementing a user interface, or GUI, to allow the users to access and use the functionality of the system. However, it was not just a simple task of writing java code. As with all implementations, the functionality structure of the system had to be designed. The user functionality was mapped and designed using the UML modeling language.   The GUI has not been fully completed as this project was a proof of concept. However, multiple user interfaces were developed to allow the system to be

accessed and tested. In Figure XVI, we can see the Application details of the system.

This is just one example of the system that was developed.

Figure XVII Application Details Form

6.4   Expectations

When we take on any piece of work, a list of expectations are drawn up with time deliverables and resources assigned to them. We have a structured list of what we plan to achieve from the given project. It is almost certainly a difficult task to deliver all the tasks given due to unforeseen issues and risks that may or may not arise. We also have the time limitations factor, resource management and development issues.

Taking these risk factors into account, it was always of the understanding that this project would be a proof of concept with the hope that when the deadlines occurred that each and every task would be successfully delivered. Throughout most of the projected timelines, especially the documentation deadlines were met, it is unfortunate that the front end development hasn't been completed as of yet. However, there is a working system in place that will be continually developed.

Moreover, the system is expected to evolve with new design considerations and development work in progress. It was expected to have around 90% of the front end coding complete at this stage. However, 60% is a more accurate figure. While it was possible to have implemented a fully working solution, this approach would have resulted in much reduced functionality. A decision was made to create a proper system rather then one that worked. It is since expected to have this fully completed and implemented by year end with the expected extra security in place and a more efficient system developed.

6.5   Future Work

It has already been decided that this project will continue to be developed in the future. In the future, it is hoped that this system will provide an important system to business environments. We will continue developing this system, and in the near future it will hopefully become a sellable product.

The next stage will be to complete the GUI front end and integrate the system into the business environment in order to fully populate and integrate the system into the process cycle. Going forward, this system will be key element to the IT application functions.

The database structure is currently very stable and efficient. The GUI needs more work to improve the "look and feel", and to provide more functionality. The lack of security entry will be implemented in due time; the development code for this has already been written, but this has not yet been tested.

In the future, it is planned to continue to develop this product as a generic product that can be adopted and implemented into any company's environment. It is intended that this will be an easy task, once the baseline development has been completed and the system has been tested and approved.

6.6   Conclusions and Recommendations

This dissertation and project area has been an exciting learning experience, from the research area to the design and development areas. The experience gained through the project has been invaluable, not only from an understanding of the current systems and process flow, but from an understanding of how people interpreted the problem idea and solution.

During the research, it was apparent that the system is needed for specific types of project areas and processes. The research area was very interesting, showing what the different tasks were and why the system was needed. We can also see a trend developing from the migration project, and we can see the benefits from the development of this system

The design and implementation phases of the project were certainly time consuming and gave us a tremendous learning experience. Designing the database structure, that has now grown to a more complex but efficient system, has certainly given us a better understanding of peoples needs and how we can deliver a solution to them. There are some more design considerations that can be integrated, but again these will be completed in time.

The idea behind developing such a broad area system was not to be as specific as the previous products have been, but to give a more general flow that can and will include a variety of technologies. We will continue to develop, and hopefully, one day market this product.

7 Works Cited

[1] "Tracking Software" Retrieved on the 21st September 2006 from http://www.macrovision.com/solutions/it/tracking/index.shtml.

[2] DevTrack Retrieved on the 12th September 2006 from http://www.hallogram.com/tedevtrack/index.html

[3] Gustavo Alonso, Divyakant Agrawal, and Amr El Abbadi. Process synchronization in workflow management systems. In 8th IEEE Symposium on parallel and Distributed Processing, October 1996. – Retrieved on the 18th April 2007 from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=60317

[4] Davenport, T. H., De Long, D. W., and Beers, M. C. "Successful Knowledge Management Projects," Sloan Management Review,

  Winter 1998, pp. 43-57.   Retrieved on the 18th September 2006 from http://www.providersedge.com/docs/km_articles/Building_Successful_KM_Projects. pdf.

[5] Quest Application Management Products – Retrieved on the 21st September 2006 from http://www.quest.com/application_management/

[6] The SoftGrid Application Virtualization Platform – Retrieved on the 04th April 2007 from http://www.softricity.com/products/softgrid.asp.

[7] Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html

 [8] Systems Development Life Cycle (SDLC) Retrieved on the 07th May 2007 from http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp

[9]   What   is   UML?   Retrieved   on   the   08th   May   2007   from http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm.

[10]   Software   Models   Retrieved   on   the   17th   May   2007   from http://www.excelsoftware.com/models.html .

[11]   The   Use   Case   Model   Retrieved   on   the   17th   May   2007   from http://www.sparxsystems.com.au/resources/tutorial/use_case_model.html .

[12]   Normalization   Retrieved   on   the   19th   May   2007   from http://www.webopedia.com/TERM/N/normalization.html.

[13] "Referential Integrity" From Mike Chapple, Your Guide to Databases. Retrieved on   the   20th   August   2007   from http://databases.about.com/cs/administration/g/refintegrity.htm

[14]   UML   2   Class   Diagrams   Retrieved   on   20th   May   2007   from http://www.agilemodeling.com/artifacts/classDiagram.htm.

[15]   Installation   Package   –   Retrieved   on   the   06th   May   2007   from http://msdn2.microsoft.com/en-us/library/aa369294.aspx.

[16] Overview of Windows Installer – Retrieved on the 06th May 2007 from http://msdn2.microsoft.com/en-us/library/aa370566.aspx.

[17] Chapter 1 - Introducing Systems Management Server Retrieved on the 10th August   2007   from   - http://www.microsoft.com/technet/prodtechnol/sms/sms2/proddocs/smsplan/smsplan. mspx?mfr=true

[18] Software Asset Review 2006 and Anti-Piracy Information Retrieved on 26 August 2007 from http://w3.bsa.org/ireland/

8 Appendixes

A. Oracle SQL Code

Drop Table Vendor_TBL;

Drop Table Lang_TBL;

Drop Table AppType_TBL;

Drop Table Lic_TBL;

Drop Table App_TBL;

Drop Table DevType_TBL;

Drop Table Dev_TBL;

Drop Table UserStat_TBL;

Drop Table Dept_TBL;

Drop Table Users;

Drop Table Status_TBL;

Drop Table Priority_TBL;

Drop Table AppPro_TBL;

Drop Table ADGroup_TBL;

Drop Table Disc_TBL;

Drop Table DeptApp_TBL;

Drop Table UserDApp;

Commit;

Drop Table Vendor_TBL;

Create Table Vendor_TBL(

Vendor_ID   Number(10) NOT NULL,

Vendor_Name   VARCHAR2(10) NOT NULL,

Contact_Name   VARCHAR2(20) NOT NULL,

Email_Add   VARCHAR2(30),

Vendor_URL   VARCHAR2(60),

Phone_No   Number(10),

Constraint PK_Vendor_ID Primary Key(Vendor_ID));

Commit;

Drop Table Lang_TBL;

Create Table Lang_TBL

Language_ID   Number(10) NOT NULL,

Language_Code   VARCHAR2(5) NOT NULL,

Language_Name VARCHAR2(10) NOT NULL,

Constraint PK_Language_ID Primary Key(Language_ID));

Commit;

Drop Table AppType_TBL;

Create Table AppType_TBL(

Type_ID   Number(10) NOT NULL,

Type   VARCHAR2(10),

Constraint PK_Type_ID Primary Key(Type_ID));

Commit;

Drop Table Lic_TBL

Create Table Lic_TBL

License_ID   Number(10) NOT NULL,

License_Type   VARCHAR2(20) NOT NULL,

License_Code   VARCHAR2(20) NOT NULL,

License_Quant   Number(10) NOT NULL,

Constraint PK_License_ID Primary Key(License_ID));

Commit;

Drop Table App_TBL;

Create Table App_TBL(

App_ID   Number(10) NOT NULL,

Vendor_ID   Number(10) NOT NULL,

Language_ID   Number(10) NOT NULL,

Type_ID   Number(10) NOT NULL,

License ID   Number(10),

App_Name   VARCHAR2(20)NOT NULL,

App_Ver   Number(10) NOT NULL,

Rev_No   Number(10) NOT NULL,

Constraint PK_App_ID Primary Key(App_ID),

Constraint   FK_Vendor_ID   Foreign   Key(Vendor_ID)   References

Vendor_TBL(Vendor_ID) On Delete Cascade,

Constraint   FK_Language_ID   Foreign   Key(Language_ID)   References

Lang_TBL(Language_ID) On Delete Cascade,

Constraint   FK_Type_ID   Foreign   Key(Type_ID)   References

AppType_TBL(Type_ID),

Constraint   FK_License_ID   Foreign   Key(License_ID)   References

Lic_TBL(License_ID));

Commit;

Drop Table DevType_TBL;

Create Table DevType_TBL(

DevType_ID   Number(10) NOT NULL,

Dtype   VARCHAR2(20) NOT NULL,

Constraint PK_DevType_ID Primary Key(DevType_ID));

Commit;

Drop Table Dev_TBL;

Create Table Dev_TBL (

Dev_ID   Number(10) NOT NULL,

Fname   VARCHAR2(20) NOT NULL,

Lname   VARCHAR2(20) NOT NULL,

DevType_ID   Number(10) NOT NULL,

Constraint PK_DEV_ID Primary Key(Dev_ID),

Constraint   FK_DevType_ID   Foreign   Key(DevType_ID)   References

DevType_TBL(DevType_ID));

Commit;

Drop Table UserStat_TBL;

Create Table UserStat_TBL(

UserS_ID Number(10) NOT NULL,

Status   VARCHAR2(10) NOT NULL,

Constraint PK_UserS_ID Primary Key(UserS_ID));

Commit;

Drop Table Dept_TBL;

Create Table Dept_TBL(

Dept_ID   Number(10) NOT NULL,

Dept_Type   VARCHAR2(20) NOT NULL,

Constraint PK_Dept_ID Primary Key(Dept_ID));

Commit;

Drop Table Users;

Create Table Users_TBL(

User_ID   Number(10) NOT NULL,

UserS_ID Number(10) NOT NULL,

Fname   VARCHAR2(20) NOT NULL,

Lname   VARCHAR2(20) NOT NULL,

Dept_ID   Number(10) NOT NULL,

User_Acc VARCHAR2(20) NOT NULL,

Email_Add   VARCHAR2(40) NOT NULL,

Phone_No   Number(10),

Location   Number(10),

Floor   Number(10),

Manager_ID   Number(10),

Job_Title VARCHAR2(20) NOT NULL,

Constraint PK_User_ID Primary Key(User_ID),

Constraint   FK_UserS_ID   Foreign   Key(UserS_ID)   References

UserStat_TBL(UserS_ID) On Delete Cascade);

Commit;

Drop Table Status_TBL;

Create Table Status_TBL(

Status_ID Number(10) NOT NULL,

Status_Type   VARCHAR2(20) NOT NULL,

Constraint PK_Status_ID Primary Key(Status_ID));

Commit;

Drop Table Priority_TBL;

Create Table Priority_TBL (

Priority_ID   Number(10) NOT NULL,

Priority_Name   VARCHAR2(20) NOT NULL,

Constraint PK_Priority_ID Primary Key(Priority_ID);

Commit;

Drop Table AppPro_TBL;

Create Table AppPro_TBL(

AppPro_ID   Number(10) NOT NULL,

App_ID   Number(10) NOT NULL,

Dev_ID   Number(10) NOT NULL,

Status_ID Number(10) NOT NULL,

Tester_ID Number(10) NOT NULL,

Priority_ID   Number(10) NOT NULL,

Request_Date   VARCHAR2(10) NOT NULL,

Test_Date VARCHAR2(10) NOT NULL,

Deploy_Date   VARCHAR2(10) NOT NULL,

Notes   VARCHAR2(60),

Constraint PK_AppPro_ID Primary Key(AppPro_ID),

Constraint FK2_App_ID Foreign Key(App_ID) References App_TBL(App_ID)

On Delete Cascade,

Constraint FK_Dev_ID Foreign Key(Dev_ID) References Dev_TBL(Dev_ID)

On Delete Cascade,

Constraint   FK_Status_ID   Foreign   Key(Status_ID)   References

Status_TBL(Status_ID) On Delete Cascade,

Constraint   FK_Tester_ID   Foreign   Key(Tester_ID)   References

Dev_TBL(Dev_ID) On Delete Cascade,

Constraint   FK_Priority_ID   Foreign   Key(Prioriry_ID)   References

Priority_TBL(Priority_ID) On Delete Cascade);

Commit;

Drop Table ADGroup_TBL

Create Table ADGroup_TBL(

ADGroup_ID   Number(10) NOT NULL,

ADGroup_Name VARCHAR2(64) NOT NULL,

Shortcut_Name   VARCHAR2(20) NOT NULL,

App_ID   Number(10) NOT NULL,

Constraint PK_ADGroup_ID Primary Key(ADGroup_ID),

Constraint FK3_App_ID Foreign Key(App_ID) References App_TBL(App_ID)

On Delete Cascade);

Commit;

Drop Table Disc_TBL;

Create Table Disc_TBL(

Disc_ID   Number(10) NOT NULL,

App_ID   Number(10) NOT NULL,

Dev_ID   Number(10) NOT NULL,

Doc_Link VARCHAR2(80) NOT NULL,

Constraint PK_Disc_ID Primary Key(Disc_ID),

Constraint FK4_App_ID Foreign Key(App_ID) References App_TBL(App_ID)

On Delete Cascade,

Constraint FK2_Dev_ID Foreign Key(Dev_ID) References Dev_TBL(Dev_ID)

On Delete Cascade);

Commit;

Drop Table DeptApp_TBL;

Create Table DeptApp_TBL(

DeptA_ID   Number(10) NOT NULL,

Dept_ID   Number(10) NOT NULL,

App_ID   Number(10) NOT NULL,

Constraint PK_DeptA_ID Primary Key(DeptA_ID),

Constraint   FK2_Dept_ID   Foreign   Key(Dept_ID)   References

Dept_TBL(Dept_ID) On Delete Cascade,

Constraint Fk5_App_ID Foreign Key(App_ID) References App_TBL(App_ID)

On Delete Cascade);

Commit;

Drop Table UserDApp;

Create Table UserDApp_TBL(

UserDApp_ID   Number(10) NOT NULL,

App_ID   Number(10) NOT NULL,

User_ID   Number(10) NOT NULL,

Constraint PK_UserDApp_ID Primary Key(UserDApp_ID),

Constraint Fk6_App_ID Foreign Key(App_ID) References App_TBL(App_ID)

On Delete Cascade

Constraint FK_User_ID Foreign Key(User_ID) References User_TBL(User_ID)

On Delete Cascade);

Commit;

B. Use Cases

Use Case – View Users
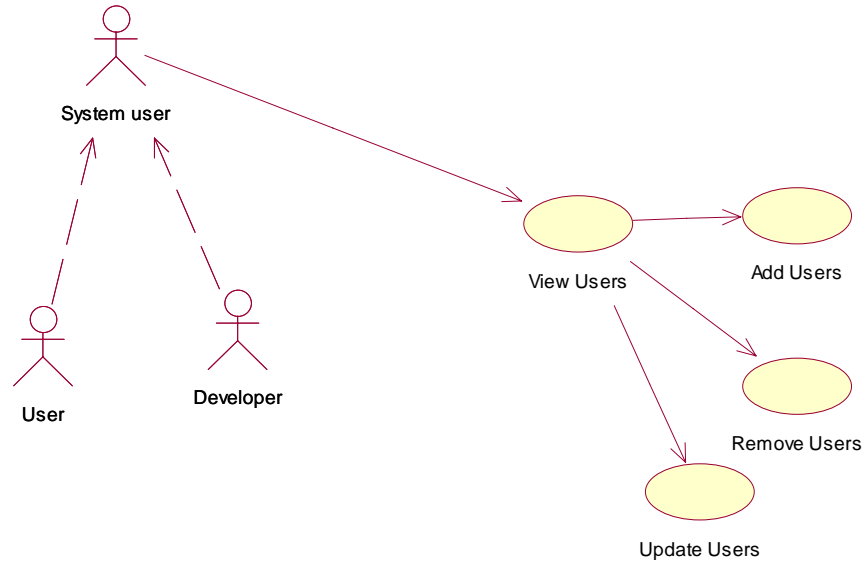
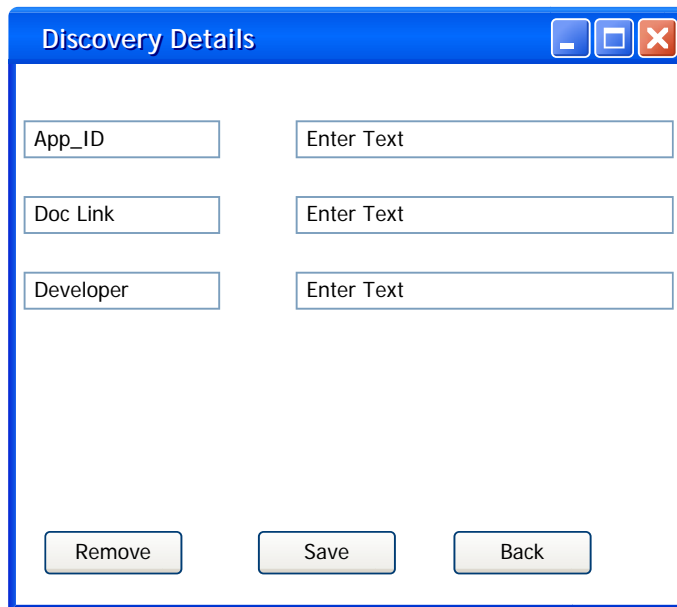Use Case – View Department
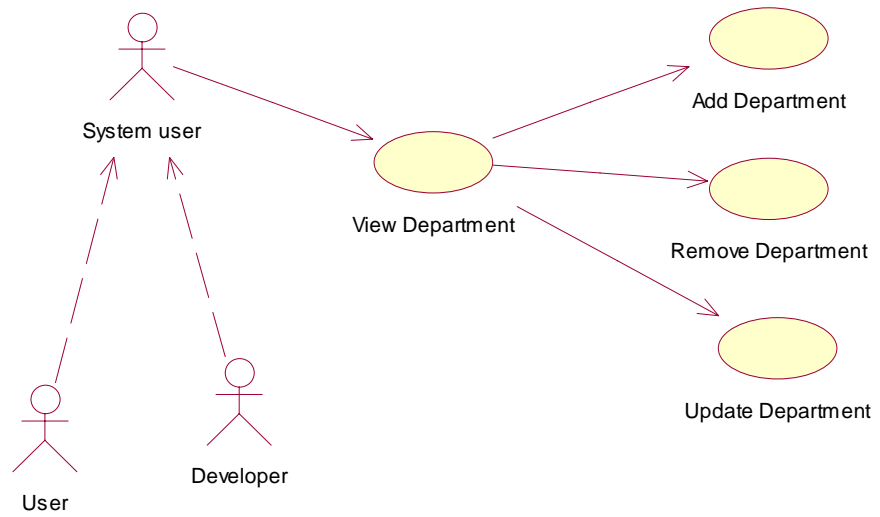
Use Case Language Details
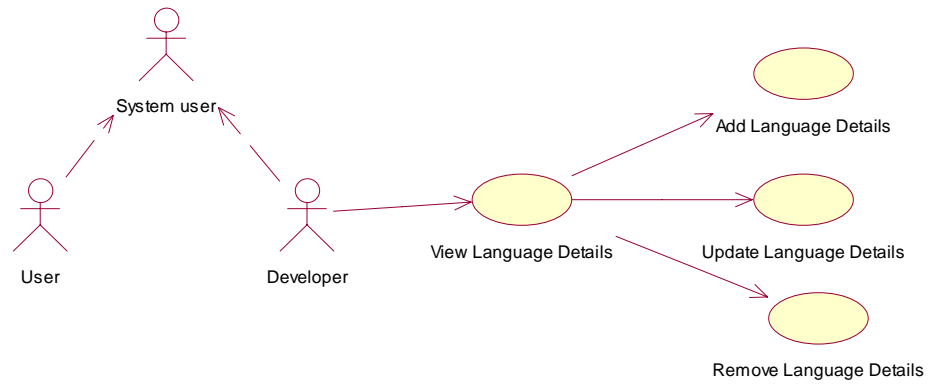
Use Case – Developer Details
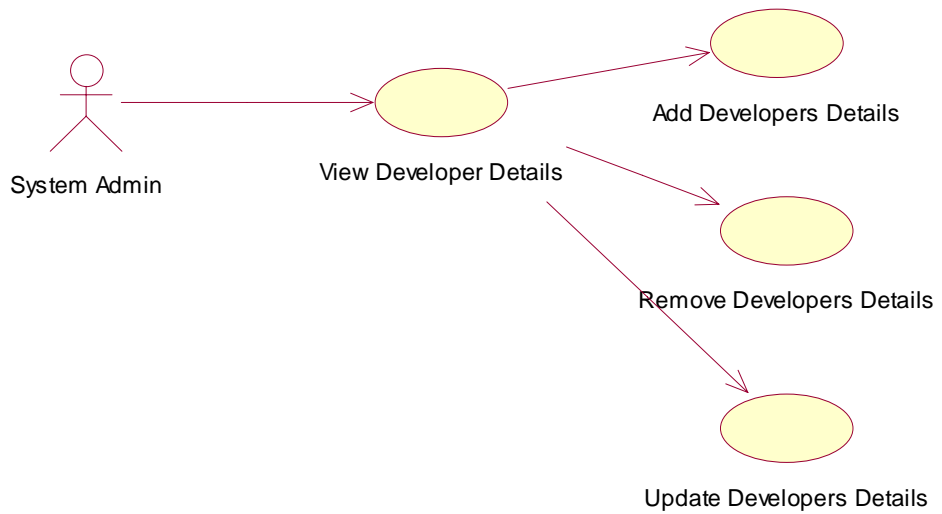
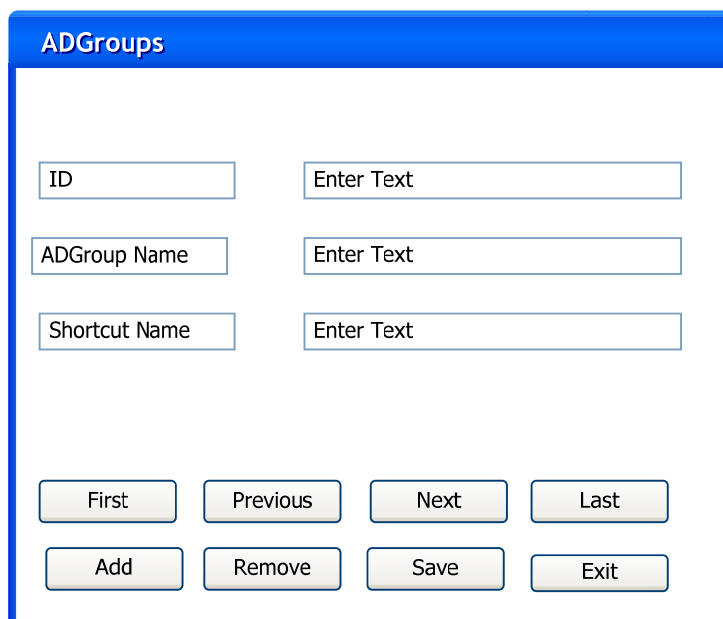## Use Case Generate Reports
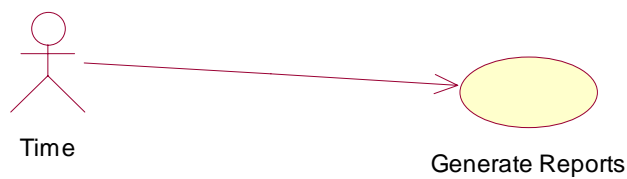
C. GUI Design

The Discovery Details displays the information captured by the developer that specifies how to install and configure the application, you will notice we can remove this detail and save (update) and also go back to the previous Application Details Screen. Once returned to the application Details Screen we can now select the ADGroups option that will display the following.

The ADGroups screen displays the details for that application there can be many or one ADGroups for that application and we again have our standard flow with exit returning us to the previous screen.

Once back in the Application Details Screen we can select Exit to return us back to the main menu. We can now take a different flow, User Details. Once selected the below screen will appear.

We can see there is not much to this one, just entering in details and check the list of users. Exit will bring us back to the main menu. Here we can select the Language Details now.

Same flow again and for the next two in the main menu.

Department Details

The Developer Details

Once the flow as returned to the main menu, we can now select exit that will now exit the application tool. Please note this is only a GUI prototype it is not the actually system, there will be differences and possible more screens available this is only to give the business and users an idea how the application might look once it is finished.

9 Annotated Bibliography:

[1] "Tracking Software" Retrieved on the 21st September 2006 from http://www.macrovision.com/solutions/it/tracking/index.shtml.   This   article summarises the need for tracking software, it is a marketing promotion for this companies product, however it outlines that such tracking systems are necessary. It is a leading product in the application development and deployment area but does not give a broad range to capture all the processes that are required for a more efficient product.

[2]   DevTrack   Retrieved   on   the   12th   September   2006   from http://www.hallogram.com/tedevtrack/index.html - This is another system that is used for application tracking. This article describes the product and what it can provide for businesses. It was very important for this dissertation to understand the competition.

[3] Gustavo Alonso, Divyakant Agrawal, and Amr El Abbadi. Process synchronization in workflow management systems. In 8th IEEE Symposium on parallel and Distributed Processing, October 1996. – Retrieved on the 18th April 2007 from   http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=60317.   This   paper discussed change management in software engineering, it specifically talks about application changes within business and how it will effect the environment. It is relevant when we are talking about the need for a tracking system and having proper research done on the area in question.

[4] Davenport, T. H., De Long, D. W., and Beers, M. C. "Successful Knowledge Management Projects," Sloan Management Review,

  Winter 1998, pp. 43-57.   Retrieved on the 18th September 2006 from http://www.providersedge.com/docs/km_articles/Building_Successful_KM_Projects.pdf. This article gives an overview on how we can build a successful knowledge management projects. It was relevant during the research of this dissertation to understand what systems have already been developed and how.

[5] Quest Application Management Products – Retrieved on the 21st September 2006 from http://www.quest.com/application_management/ .   This website gives an overview of other application management products. It is important during this dissertation to understand what toolsets are currently available and why they can or cannot be suited towards what we are trying to achieve.

[6] The SoftGrid Application Virtualization Platform – Retrieved on the 04th April 2007 from http://www.softricity.com/products/softgrid.asp. This article gives us an overview of the SoftGrid technology, giving us a better understanding of the application delivery technology. This is relevant during the research period.

[7] Quality Control and Software Process models – Software Engineering 8th Edition by Ian Sommerville. Retrieved on the 10th January 2007 from http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/Presentations/index.html .   This book covers a wide range of topics in software engineering. The main ones we focused on were the quality control and software processes. The attached website is in PowerPoint format, we obtained diagrams and data from this source.

[8] Systems Development Life Cycle (SDLC) Retrieved on the 07th May 2007 from http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp. In this article it outlines the SDLC giving a general overview of the cycle. This was important when discussing the software cycle with relevance to the model we would choose for our system.

[9]   What   is   UML?   Retrieved   on   the   08th   May   2007   from http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm. This gives a brief understanding and view into UML explaining to the reader what UML is and how we can use this. This article was used to summarise UML so we could understand how we can design different angles of our system

[10]   Software   Models   Retrieved   on   the   17th   May   2007   from http://www.excelsoftware.com/models.html . This website gives us a breakdown of the different UML software models we can use. It was very useful to obtain this for a more detailed understanding into what UML is during the writing of this dissertation.

[11]   The   Use   Case   Model   Retrieved   on   the   17th   May   2007   from http://www.sparxsystems.com.au/resources/tutorial/use_case_model.html   .   This article develops on Use Case models and gives the reader a better understanding of each element needed and used in Use Case diagrams.

[12]   Normalization   Retrieved   on   the   19th   May   2007   from http://www.webopedia.com/TERM/N/normalization.html. Normalisation is a topic that is well known throughout relational database structures. When discussing

databases we looked towards this article to better explain and understand what normalisation is used for and to give a brief overview how we can apply it. It is very relevant in designing and developing the database structure.

[13] "Referential Integrity" From Mike Chapple, Your Guide to Databases. Retrieved on the 20th August 2007 from http://databases.about.com/cs/administration/g/refintegrity.htm. Referential integrity is a relational database methodology that is implemented to make a database structure relational so they can link into and manage the relationships correctly. This gives a brief outlining into what referential integrity is and where it is used.

[14] UML 2 Class Diagrams Retrieved on 20th May 2007 from http://www.agilemodeling.com/artifacts/classDiagram.htm. This again is another article that describes UML 2 class diagrams, used to give a fuller background on the understanding and usage of UML 2 class diagrams. This is used to develop and design the code when developing the GUI.

[15] Installation Package – Retrieved on the 06th May 2007 from http://msdn2.microsoft.com/en-us/library/aa369294.aspx. We have discussed topics on MSI and windows installer, this article is a brief outline on what an MSI is and where we use it.

[16] Overview of Windows Installer – Retrieved on the 06th May 2007 from http://msdn2.microsoft.com/en-us/library/aa370566.aspx. This source was used to

give an overview on what Microsoft Windows Installer is and how we can develop it and why we use it.

[17] Chapter 1 - Introducing Systems Management Server Retrieved on the 10th August 2007 from - http://www.microsoft.com/technet/prodtechnol/sms/sms2/proddocs/smsplan/smsplan.mspx?mfr=true . SMS is a method of application deployment. This article describes how we can deliver applications to user's desktops and how it can do so much more in the application management area. To gain a better understanding and give an overview this article digs deeper into how we can utilise from this product.

[18] Applied Cryptography 2nd Edition. By Bruce Schneier, Counterpane Labs. Retrieved on the 25th May 2007. This book gives an in-depth knowledge of cryptography. It is used and will be used to understand security aspect to the system and will develop a secure system from the research and knowledge gained from this book. It is important to this dissertation as we intended to apply security to this system in the near future.

[19] Java Coding references – Retrieved on the 11th November 2006 from http://java.sun.com. This website is an important reference point that references different coding classes and helped during the development of this project. It has an in depth knowledge of coding techniques that were used.

[20] The Complete Reference SQL By: James R. Groff & Paul N Weinberg – Retrieved 12th September 2006. This book is an excellent SQL reference and

develops on the knowledge of SQL techniques used to design and develop the database structure.

[21] Software Asset Review 2006 and Anti-Piracy Information Retrieved on 26 August 2007 from http://w3.bsa.org/ireland/. This website discusses the anti-piracy laws and is designed to track down business to audit their systems to ensure they have the correct amount of license for their applications.