

**Examination Request System for the
Clinical Radiology dept. at Leeds
General Infirmary**
Sean Webster
BSc Information Systems
Session (2005/2006)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material, which is obtained from another source, may be considered as plagiarism.

(Signature of student)_____

Summary

The main aim of this project was to create a system, to at least a prototype standard, which allows qualified members of departmental staff to request a medical examination electronically opposed to manually using the current card-based system. The intention was to be able to provide a system, which gives users an enhanced interface for the requesting of examinations, and to offer more support in the process of selecting an examination to improve accuracy and save time.

Background research was conducted before any implementation was performed, and it was evident that there would be some definite changes to business process as a result of creating the system. The analysis chapter focuses heavily on the collection of general and user requirements and details all the different processes currently involved with the manual-system, illustrating them using a variety of different business process diagrams. The existing processes are also compared with the process changes the new system would incur; the results of which were used and analysed further in a report created for the department management.

The prototype was then designed and implemented according to the requirements gathered and tested thoroughly to eliminate any coding errors. The system was evaluated to ensure it met all the objectives and user requirements and to assess the level of which the new system provides improved quality of information for system users, through ease of use, accuracy and the saving of time with comparisons drawn from the use of the current card-based system.

The project was successful in that the aim, objectives and minimum requirements were all met and exceeded. The system was only required to be to prototype standard; and as a result would need some changes before it could be integrated into current practices, in particular the clinical support interface needs further expansion and the database connectivity code would need to be changed to allow the system to connect to the departments database, however both these changes have been explained and detailed to the management in the system user-guide to make integration easier.

Acknowledgements

I would like to take time out to thank everyone who has helped me in the completion of this project. Firstly, I would like to thank Dr. Lydia Lau, who lent me the relevant business process literature, which enabled me to complete the analysis section of this project.

I would also like to thank my supervisor Dr. Stuart Roberts, who has helped look over my work at various stages and help guide me in the right direction. Furthermore, I would like to thank my assessor Dr. Kevin McEvoy for his advice and feedback during the project.

Finally, and most importantly, I would like to thank everyone at the Department of Radiology at Leeds General Infirmary, without their input, support and continuous information this project would not exist.

Contents

CHAPTER 1: Introduction		[Page]
1.1 Problem Overview	1
1.2 Project Aim	2
1.3 Project Objectives	2
1.3.1 Minimum Requirements	2
1.3.2 Deliverables	3
1.4 Project Schedule	3
CHAPTER 2: Background Research		
2.1 Background to Problem	4
2.2 Methodology Research	5
2.2.1 The Waterfall Model	6
2.2.2 The Spiral Model	7
2.2.3 Prototyping	8
2.3 Methodology Selection	8
2.4 Appropriate Technologies	9
CHAPTER 3: Analysis		
3.1 Analysis of current processes	11
3.2 Requirements	12
3.2.1 General Requirements	12
3.2.2 User Requirements	12
3.3 BPM using UML	13
3.3.1 High-Level Activity Diagram	14
3.3.2 Business Object Model	15
3.3.3 Business Use-Case Diagram	16
3.3.4 Low-Level Activity Diagram	17

CHAPTER 4: Design

4.1 Introduction	18
4.2 Design Techniques	18
4.2.1 Design Walkthrough	18
4.3 GUI design	19
4.3.1 GUI Components	20
4.3.2 Help Information	21
4.4 System Usability	21
4.4.1 Use of Colour	22
4.4.2 Form Designs	22
4.5 Database Design	25
4.6 E-R Modelling	25
4.7 Table Design	26
4.8 Complete E-R Diagram	27

CHAPTER 5: Implementation

5.1 Introduction	28
5.2 The Technical Platform	28
5.3 GUI Implementation	29
5.4 Clinical Help Implementation	30
5.5 Setting up the database	31
5.6 Connecting to the database	32
5.6.1 Driver Importance	33
5.7 Writing and executing the SQL	33
5.8 Processing the results	34
5.9 Problems Encountered	35
5.10 Screen Sources	35

CHAPTER 6: Testing

6.1 The need for testing	36
6.2 Types of testing	36
6.3 Code Testing	36
6.4 System Testing	37
6.5 Functionality Testing	37
6.6 Meeting User Requirements	39
6.7 User Walkthrough	39
6.7.1 Login Screen	40
6.7.2 Exam Request GUI	40
6.7.3 Clinical Support Screen	40

CHAPTER 7: Evaluation

7.1 Introduction	41
7.2 Information Gathering	41
7.3 Determining Evaluation Objectives	42
7.4 Choosing Evaluation Paradigm	42
7.5 Usability Testing	43
7.5.1 Test Users	43
7.5.2 Performing the Test	44
7.5.3 Usability Test Results	44
7.6 Heuristic Evaluation	45
7.6.1 Expert Users	46
7.6.2 Performing the test	46
7.6.3 Heuristic Evaluation	46
7.7 Card-based system v electronic system	48
7.8 Meeting Requirements	48
7.9 Evaluating User-Guide	49
7.10 Future Improvements	50

7.11 Conclusion	50
References.....	51
Appendix A – Personal Reflection.....	53
Appendix B – Project Schedule.....	55
Appendix C – Complete E-R Diagram.....	57
Appendix D – Business Process Report.....	59
Appendix E – Testing.....	70
Appendix F – User Guide.....	75
Appendix G – Usability Testing.....	99
Appendix H – Heuristic Evaluation.....	107
Appendix I – Screen Sources.....	110

Chapter 1

Introduction

1.1 Problem Overview

In the department of Clinical Radiology at Leeds General Infirmary (L.G.I), if a patient requires an examination, a qualified member of staff fills out an examinations request card, which then has to be verified by a doctor, who in-turn decides whether the examination is appropriate and specifies which one is required. The current system is inefficient, time-consuming and can lead to lost requests.

Requests for different types of examinations such as Radiology, Ultrasound, CT scans and MRI scans need to be authorised by a doctor. Currently requests for examinations are submitted on a hand written card. These are then vetted for appropriateness by a doctor and booked in if they are to go ahead. The request is then scanned and stored digitally. However, it is seen that an electronic requesting system could be implemented.

There are reasons for dissatisfaction with the current system. Some of the current problems with the manual system include lost requests, as cards are relatively small and as they can pass through a number of people and areas in the department, potentially, they can be lost or misplaced. There are also problems with duplicate requests and internal mail movement, if a card is in circulation there is no way of knowing of its existence until it has been stored digitally. Furthermore, there are security issues to consider, request cards can be potentially stolen or looked at by unauthorised personnel, the request card holds sensitive patient data and this is not fully protected with the manual system.

Finally, the request card allows for a consultant to be assigned to a patient, however if the patient returns for another examination the consultant may be different to the one originally on the card.

Currently stickers are used to place over the old consultant to update it, however it has been known for the stickers to peel off in the internal mail system, causing problems such as delays and wrong assignment.

1.2 Aim

The aim of this project is to create a system, to at least prototype standard, which simulates and carries out the requesting of examinations. On completion of the system, users will have an enhanced interface for requesting examinations and also more support to aid in examination selection to improve accuracy and save time.

1.3 Objectives

- ❖ To develop and fully test a new system, including any new processes and/or IT artefacts for capturing, authorising and communicating requests for radiological examinations.
- ❖ To integrate the new system into existing procedures and technologies, ensuring the technical accommodation of existing practices.
- ❖ To provide training materials for users of the new system, including a manual and user-documentation detailing system use.
- ❖ To model current business processes using established business-modelling techniques.

1.3.1 Minimum Requirements

- A system GUI which:
 - Allows for the editing and changing of examination details.
 - Provides improved functionality, for clinical information, to help in the selecting of examinations.
 - Has an appropriate security measure in place to prevent unauthorised access.
 - Can support the electronic requesting of examinations.
- A model of the problem domain using established business-process modelling techniques, to form the basis of a short report, presenting the current business processes and a comparison with the new process changes from the implementation of the electronic requesting-system
- A series of built automated tests, which simulate user behaviour and verify the results.

- A document for management and staff, detailing to managers how the system could be integrated into current practices, and documenting the systems appropriate use, to staff and all potential users within the department.
- An evaluation of the system in terms of improved quality of information.

The possible extensions are:

- A facility, which provides advice, electronically, on the choice of procedure, allowing a qualified member of staff to complete the digital examinations request form without the need to confirm with a doctor.

1.3.2 Deliverables

The deliverables for this project are therefore:

- A Project Report.
- A Business-Process report for departmental management, detailing all the process changes.
- A prototype to be used for demonstration and user feedback.
- A working prototype system accompanied with a user-guide to allow hand-over to the department.

The prototype will be a first build of the system, there will be the relevant documentation in the user-guide which will be presented to management which will allow enhancements and improvements to be made as well as the changes which will be required to turn the prototype into a fully functional system which the department can integrate into their practices.

1.4 Project Schedule

The full project schedule can be viewed in Appendix B at the end of the report. The schedule was reviewed regularly to ensure deadlines were met; the schedule was important due to the size of the project and the limited time available to complete it, therefore my time had to be accurately managed. The background research and reading was completed in the first semester of 2005 with the main implementation and write-up work balanced out during the whole second semester of 2006. The schedule was altered slightly once as the work, which was predicted for completion, was underway and fully realised, the revisions made time management more realistic.

Chapter 2

Background Research

2.1 Background to the problem.

Before work can commence on a system to solve the problem, background research has to be thoroughly conducted. In order to fully understand the problem I took it upon myself to meet regularly with the problem holder and people who were involved, daily, with the current system. The purpose of the meetings was to gather as much data as possible to explore the current procedures and discuss any associated problems experienced. The information which has been collated has allowed me to progress and begin to create the user-requirements for a new system; using it as a starting point for system design.

A face-to-face meeting has the advantage of being bi-directional and collaborative; it allows for extra questions to be put-forward and also allows for clarification of un-clear answers, according to Pincus (2004). It has the obvious advantage over e-mail and message boards, in that it is in real-time and there is no delay involved. A meeting can be either formal or informal, depending on the actual situation. My initial meetings were very formal, I adopted this attitude to make sure I got as much information as possible and I also wanted to convey that I was very serious about my work. Some of my later meetings were less formal as a rapport was built with the departmental manager and staff; informal, spot, interviews were conducted to get a snapshot of departmental activities.

The General Manager is the problem owner as he employs the staff, which uses the system, and he has the authority to declare any change necessary within the department. He intrinsically understands the processes that occur in the department and the procedures involved in the current system; this qualifies him as an ideal primary source for information gathering.

Results of the meeting included detailed, descriptive documentation of the current system procedures, along with diagrams of the processes, in pictorial form, using standard business-process modelling

techniques. Resulting from this, analysis of the procedures was produced, which will be included in Chapter 3 of this report. The analysis highlights the current problems with the system; one problem highlighted is the fact that the procedures involve a mixture of paper-based resources and digital scanning. The movement of data from one medium to the other causes both inefficiency and time delays, which is a major concern when stringent deadlines have to be met. To alleviate the problem at both points it has been decided that a completely electronic based system would be beneficial, in a number of ways. The proposed solution would ideally simplify the current tasks and processes and increase departmental efficiency, where time can be spent doing things elsewhere; the time saving by the change in business process is covered in Chapter 3.

It was deduced from the meetings that a proposed system would need to be compatible and integrate with existing technology, which the department have only just recently upgraded. There are currently three systems in place which run on the departments computers; a Radiology Information System (RIS), which records patient attendance in radiology, i.e. what examination they have had. A Trust Patient Admin System, that records patient's admission to hospital, or attendance in outpatients, and also a PACS which stores the images produced from the scanning in of examination request cards – this is currently linked to RIS. The manager suggested that he would like to extend my work on a proposed system in the future so that there is no need for the PACS system or the RIS system as both could merge together, providing more efficiency and eliminating circumlocution, duplication and redundant data.

As such, the research into the technology of RIS and PACS was essential. Sources of the information gathered, were from departmental staff including radiographers, radiologists, nurses and helpers, as they were the main individuals who have practical experience and knowledge of the systems. The meetings formed relevant documentation of a list of tables currently used in the RIS system, the list was useful to form a rough idea of the database architecture and how my prototyped system could be possibly extended in the future to fully support patient data. Other information gathered from the meeting included that the database management system Microsoft SQL Server is used for the RIS and SQL used to access the database. This information proved to be useful when considering the choices of available technology to use for the proposed system.

2.2 Methodologies for Systems Development

It is important to decide from the outset, on an appropriate methodology to undertake. There does not exist a single methodology, which could be applied in full, that contains all the relevant tools and techniques, which suit a particular project. Avison & Fitzgerald (1995) state, "*The search for a perfect methodology, it is argued, is somewhat illusory*". Hence it makes sense to understand the available methodologies and choose either one, or a permutation of methodologies, to facilitate the most appropriate characteristics, and complement it with relevant tools and techniques. Choosing the correct methodology to create the system is especially important. The methodology will decide on how to characterise the system and, ultimately, set about designing it. In wrongly selecting a methodology there

is the likelihood that there will be problems during the design and implementation phases. Subsequently, a number of different methodologies are discussed below with the intention of identifying the most suitable one for this project. Due to this being a systems development project, only the methodologies from this category have been discussed leaving aside the software development methodologies such as Rational Unified Process (RUP), Rapid Application Development (RAD) and other such methodologies.

2.2.1 The Waterfall Model.

The waterfall model is a version of the systems development life cycle for software engineering. According to Beauchemin (2000), the model was the first in systems development to have a structure. It is often considered to be the “classic” approach to the systems development life cycle; it describes a development method, which is both linear and sequential. The origin of the name is explained by Fowler (2005). In a waterfall the water has no choice but to fall once it’s over the edge, the same can be said of the waterfall development, once one stage is complete it goes on to the next, the direction is one way and not bi-directional, therefore there is no turning back. The diagram below shows the basic structure of the waterfall model, the arrows represent the flow of progress from the initial collection of Requirements, through to the Evaluation:

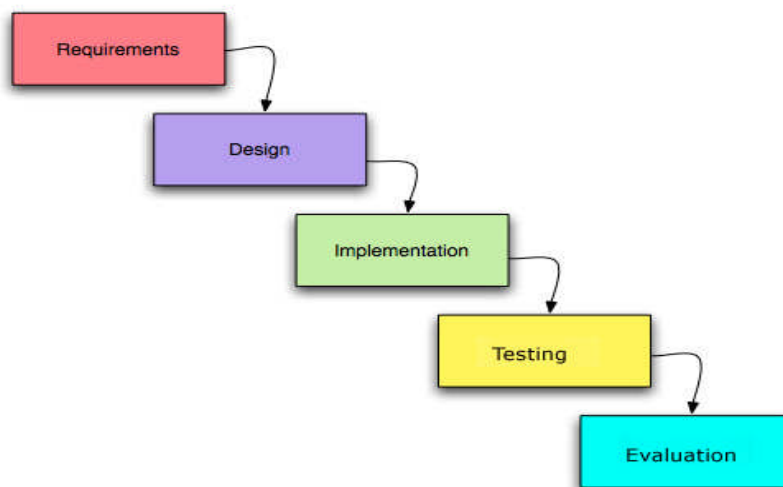


Figure 2.1: Stages in the Waterfall model.

The model has advantages and disadvantages. According to Fowler (2005) there are three main advantages; one being that testing is inbuilt to every phase in the waterfall model, so it can be applied at every point in the model to ensure it has been done correctly. Another advantage is the models enforced, disciplined approach, this allows for a schedule with deadlines to be set, these deadlines must be met allowing a timescale of the project to be allocated. The third advantage is that the model is documentation driven, which in-effect means that documentation needs to be produced at every stage in the cycle, this allows a thorough progress report throughout the cycle’s lifetime indicating where things are going right and where things may be going wrong. However, every methodology also has associated disadvantages; the waterfall model is no exception. The main disadvantage, outlined by Fowler (2005),

is “... projects rarely follow its sequential flow. This is due to the inherent problems associated with its rigid format” One example is that if a person only has vague ideas of what is exactly required from the software product, the Waterfall Model has trouble in accommodating the perfect natural occurrence of *uncertainty* at the start of the project. The waterfall model appears to be very pigeon holed in terms of its stages and it doesn’t allow phases to be completed in parallel or have the bi-directional ability to correct a stage, this is from a personal point of view.

2.2.2 The Spiral Model.

The idea here is evolutionary development, using the waterfall model for every step; this is intended to help manage risks. Beauchemin (2000) explains that the spiral model uses the method of repeatedly modifying the system until it has reached the intended clients expectations. The risk assessment is a unique inclusion to the model, which assesses the risks involved and indicates whether or not it is wise to progress. The Spiral Model is seen to be good for large and complex projects, but not as good or as effective for smaller, simpler ones.

According to, Boehm (1998), the spiral model has four important features:

- Risk analysis – Evaluates the different approaches using risk analysis. If the risk is worth continuing, an appropriate approach is identified.
- Prototyping – The prototypes are carried out by engineers, there can be many different iterations of the prototype.
- Iterative Framework – allows for ideas to be checked and evaluated.
- Alternatives – model explicitly encourages alternatives, so that the developer doesn’t become boxed in.

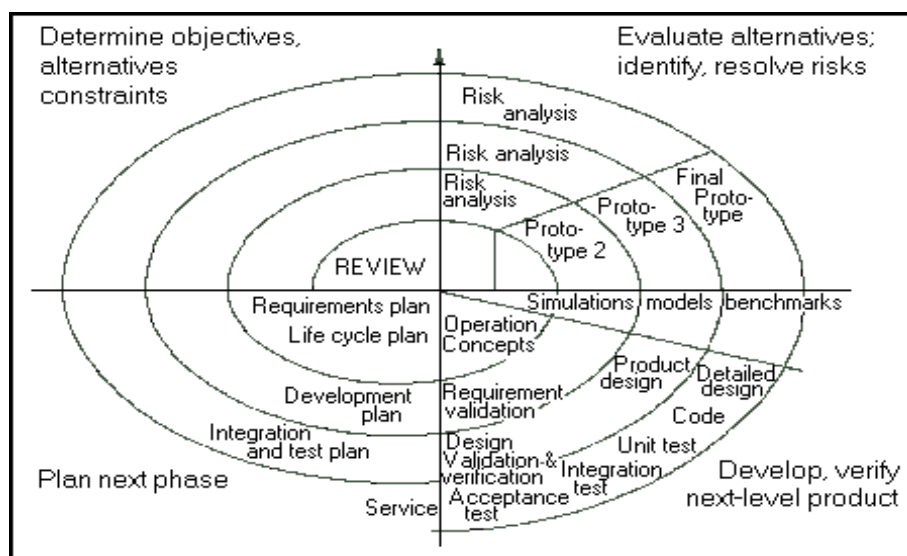


Figure 2.2: The Spiral Model, attributed to Boehm (1998).

2.2.3 Prototyping.

Beauchemin (2000) advises to use the prototyping model, if system requirements are not fully realised. Prototyping involves the creation of a prototype for a system, allowing users to assess it by using it, then giving it back to the developer to collect feedback and to further their understanding of the system requirements. The feedback can then be used to improve upon the existing prototype; this process continues in a cycle until the requirements are fully realised, allowing the system to be implemented. Figure 2.3, below, shows the most important stages involved in prototyping, it is not as stringent as the waterfall model in the sense that stages in development can be re-visited, allowing for improvements to be made. Beauchemin (2000) warns developers that by developing a system too prematurely can in-effect lead to bad choices and ineffective design, although prototyping is a rapid software development model to validate requirements, time and effort needs to be taken. With the use of this model, customers may be under the false illusion that the system will be produced sooner than it actually will be.

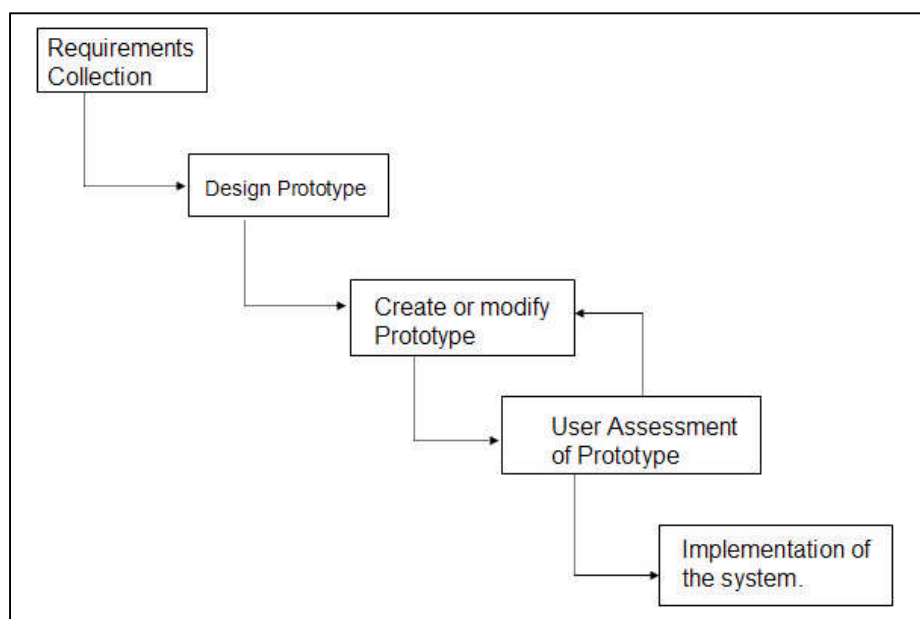


Figure 2.3: The important steps in Prototyping.

2.3 Selecting an appropriate methodology.

It is appropriate to evaluate each methodology to gain a clearer understanding of their advantages and disadvantages, and choose an appropriate one for my project. It has been already established that business-process modelling will be used to model the problem domain and capture the processes involved in the current system, to gain a snapshot of the systems functionality and people involved. However, after analysing the processes the results will be used to advance with a more complex methodology.

The three models both have distinct advantages and disadvantages, and as such a decision is difficult to make. The waterfall model is the most attractive methodology, providing requirements have been collated very early on in the project, and also providing the problem is fully understood, this model does not allow for much error, therefore it is important to get requirements right; if an implemented system is not to the users satisfaction or standard, it is very difficult to improve upon as a lot of the stages have been committed.

The prototype method can be a very tedious and time consuming model to follow; prototypes require several iterations before they are deemed to-standard and appropriate, however there is debate as to whether the changes being made are *really* what the user wants and not just being made for the sake of it. Therefore, it is probably unwise to choose this methodology as it is not a project of a large scale and time is critical.

The spiral method, combines features from both the waterfall model and prototyping, it includes repeated improvement of a system, and also considers the risks involved. However, this model requires that the system design be created relatively quickly, and it is sometimes easy for a developer to rush this through to get into the implementation phase. I am not prepared to be drawn-in by rushing, therefore this methodology seems inappropriate.

Therefore, considering the time-constraints for the project and the need to iteratively progress, it seems sensible to use the waterfall model. It ensures that work sections are done iteratively, and as such will allow the stages to be properly scheduled. By carrying out the business processing, ensures that the analysis of the system is fully completed and the problem domain is fully scoped, before progressing with the rest of the methodology, this will hopefully reduce any chance of the system not meeting user's requirements.

2.4 Appropriate Technologies

There are various means and methods for storing the required data; following interviews with the departmental manager he specified that the current architecture uses a database which stores patient information; and as such a solution would be more beneficial if it could support the integration with the database, even if not directly set-up.

An object-oriented programming approach has been deemed appropriate, as Java and C++ in particular are able to support access to the existing database. Object-oriented programming, according to Wikipedia, *"is claimed to give more flexibility, easing changes to programs, and is widely popular in large scale software engineering."*

Java is an object-oriented programming language designed to be platform independent. According

to Bennett et al (1999) “*The main idea behind this is the Java Virtual Machine involving compiling Java code to byte-code, which unlike other programming languages such as C++ it is a universal format which is interpreted in a virtual environment*”. This means that Java code can be executed on Microsoft platforms and UNIX, as well as others; this is appropriate as some of the machines within the General Infirmary are UNIX machines; therefore a Java approach would ensure full compatibility. Java is popular for creating web-applications and distributed applications, my solution will be distributed to run on existing client/server architecture; however there may be web-support, in future system builds. The Java language, does borrow from the C language, so the two approaches do have their similarities, however Astrachan et al (1999) states that Java is superior to C++ in type safety, simplicity and object-orientation.

The re-usable aspect of Java means objects can be altered, and classes re-used, which would ensure a quicker implementation and as time-constraints are precious this approach would seem appropriate. There are lots of development environments which exist to aid the process of code re-use, as code can be easily copied across to other classes within the environment, one such example is Eclipse, which gives developers “*a portable and customisable platform*” according to Aniszczyk (2003). Eclipse also indicates where code syntax is incorrect, identifying any unclosed brackets to close a method; this saves time scanning through hundreds and thousands of lines of code.

Chapter 3

Analysis

3.1 Analysis of current processes

The department of Clinical Radiology has many different types of business worker, each with their own different responsibilities and duties; however they all work together, collaboratively, to help each other complete tasks. The Examination Request system is no exception; the process of actually filling out the examination request form to it actually being approved and carried out is a long process, which involves a number of people and different permutations.

The examination request cards are, usually, filled in wherever the patient is seen, this is typically, Accident and Emergency, Outpatient departments, Wards and GP surgeries. There are others, but these are the most common areas. The request cards are received, in a few ways again usually depending on where the patient is seen. For Accident and Emergency, and some outpatients, the patient will often bring the request card to x-ray with them.

Some outpatient requests will be sent from the clinic to the department via the internal post system, as will some ward requests. GP's are able to fill out the examination request card, and use the normal post system. Several examination requests arrive by fax, this is determined upon whether a doctor is not currently available or if the request has been fast tracked, for which there can be a number of reasons.

The delay of the time it actually takes until checking and confirming an examination request, or rejecting as the case may be, obviously varies considerably, the Accident and Emergency requests will be instant as the request cards are brought in directly, or when the patient arrives in x-ray, due

to the immediate nature of the request. The most extreme cases may take 3 to 4 working days to arrive; this would only be if the examination request is deemed a low-priority, it may also be that the examination request gets re-routed through different departments before it ends up within the departments hands.

Most commonly it is a doctor who is able to confirm an examination, all grades and type of doctor are allowed by law to confirm requested x-rays; however Nurses, Radiographers and some physios / podiatrists are also allowed to authorise requests under certain conditions.

3.2 Requirements

Resulting from the meeting with the departmental manager, it became clear that the current system is entirely manual and paper based, with the exception of the digital scanning and storing of the request card. There is a Microsoft SQL Server database holding patient information, which can be accessed within the department; however they are not referenced for examination requests at the radiology side. It has been identified that a proposed prototype, as well as satisfying the requirements of requesting examinations, could be extended in the future to provide better interoperability between hospital departments. As such, these factors have been taken into consideration and a list of general and user requirements have been proposed and agreed upon with the departmental manager, taking into account the different users of the system, they are as follows:

3.2.1 General Requirements:

- I. The system should allow access to examination records and staff have the ability to edit/modify them accordingly
- II. System needs to be able to support as much clinical information as is needed for the patient.
- III. For security purposes, access to the system should be only for registered and authorised personnel.
- IV. System should allow examinations to be requested from any designated patient entry point and received at radiology ready for patient examination.
- V. System should be built in a way, which reduces the chance of erroneous or duplicate data being entered and stored into the database.

3.2.2 User Requirements:

The requirements have been split up to incorporate different requirements for different users of the system.

Nurses, Radiographers and qualified staff:

- 1) Be able to enter patient details.
- 2) Assign a patient to a consultant.
- 3) Specify the Urgency of the request
- 4) Enter clinical information about the patient.
- 5) Use clinical information and symptoms as a guide to look-up and to decide on possible examination required, this is if the user is not a doctor.

Doctors:

1. Needs to be able to do all of above, if filling in the request.
2. Needs to confirm any examination requests, which have not been decided, as such there should be a feature that allows a doctor to confirm the examination.

Manager:

- 1) Needs to be able to export the data to file to use for report writing and other statistical and management information.

3.3 Business Process Modelling: Analysing using UML

UML is an object-oriented visual modelling notation, which has nine different diagram types. *“Each diagram shows a specific static or dynamic aspect of the system”* states Erikson, Penker (2000). UML is not restricted purely to modelling processes, it can use used to good effect to model a system. However, there are many advantages of actually using UML to model business processes, UML has a standard notation which means that the tools are already in-place and are the same tools used to actually model the information systems and can be adapted and used to describe the business models. Furthermore, using an object-oriented modelling technique, which UML essentially is, has the advantage of models that are produced representing real domains and using the same terminology as the real domain. This allows models to become understandable to people who have little or no experience of UML. If the department wishes to extend upon the system in the future in the object-oriented language which it will be created in, it would be easier to create new code without furthering investigation as the models and processes will already be produced.

3.3.1 High Level Activity Diagram

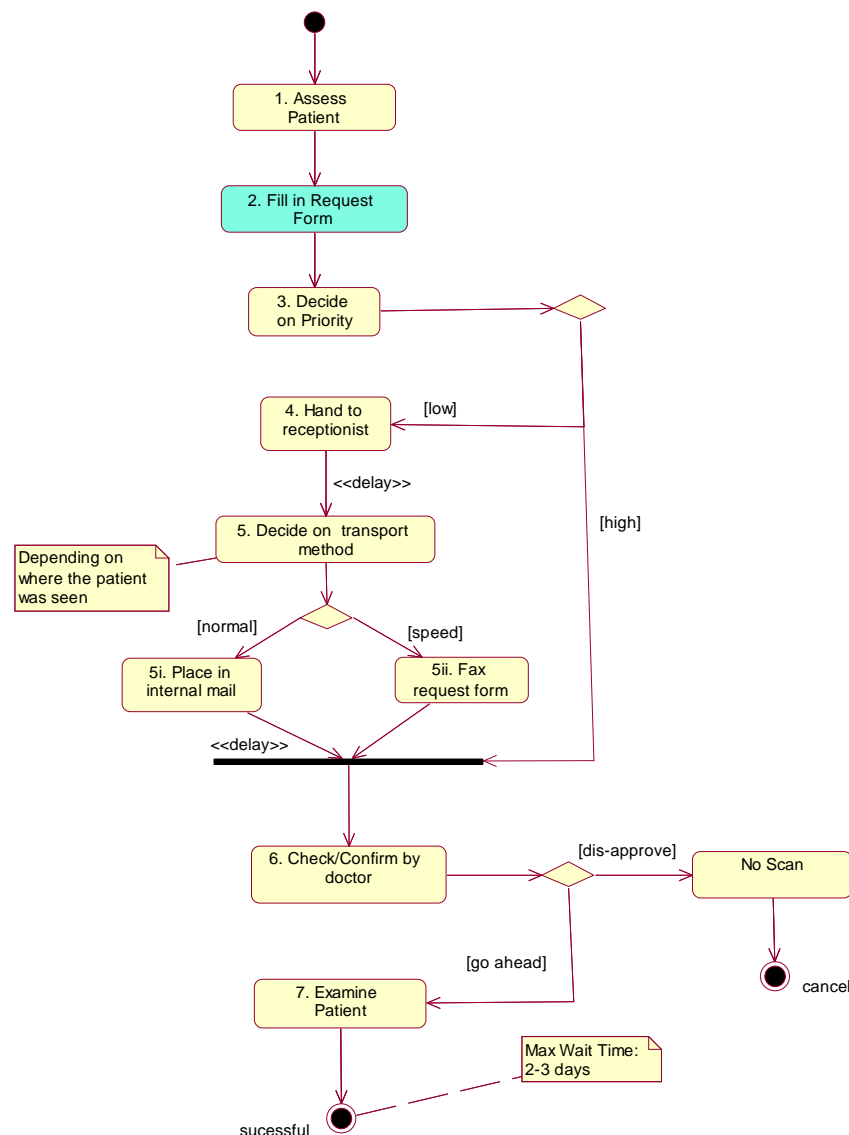


Figure 3.1: High-Level Activity Diagram of the Exam Request Process.

Figure 3.1 shows the process flow, from a high level, of the examination request card being filled in following assessment by medical staff; the steps have been numbered for reference purposes. The diagram shows that all requests go through the full sequence of processes 1-5, before a decision is made on whether the examination is to go ahead. This is not necessarily an efficient system, as time is lost getting to process number 6 if an examination is decided not to go ahead. A more efficient system may not go as far through the sample pathway. Activity 2 has been highlighted to indicate the main process, which is actually filling out the request form. Also indicated on the diagram are the delays, of which there are two. The first delay is the receptionist choosing the appropriate route and transport method of the request card, this only occurs if the examination request has been deemed a low priority, there can

sometimes be a back-log of low-priority request cards hence a delay in the actual sending. The second delay, which is marked, represents the delay experienced by the movement of internal-mail; this can sometimes take a number of days to actually reach a doctor in the worst case. There is no, *real*, delay if the transport method is by fax in the best case, requests can be usually received by the doctor in a number of minutes, providing he or she is in their office at the time of receipt.

3.3.2 Business Object Model.

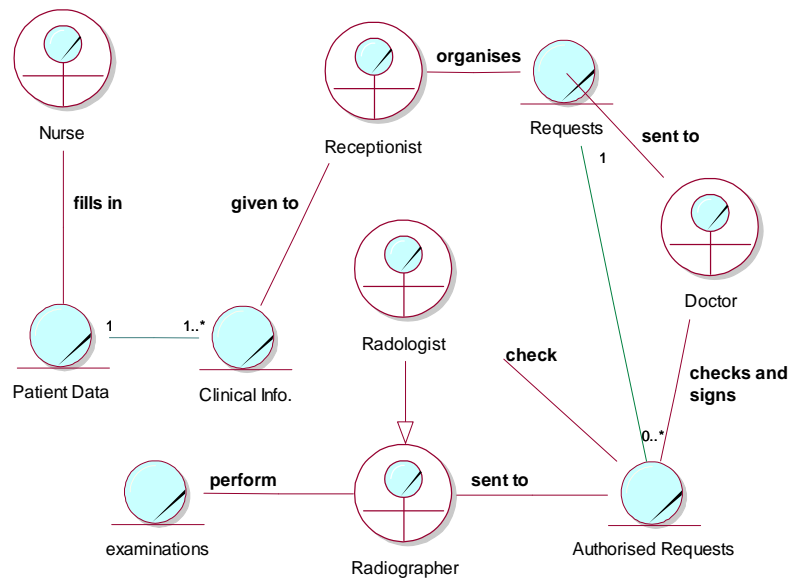


Figure 3.2: Business Object Model of the business workers and processes involved

Figure 3.2 details the interactions which take place between the business workers and the business objects they are associated with. The diagram shows a Nurse filling in patient data, and the clinical information relating to that patient, the multiplicity between the two are highlighted by a green association line, the relationship is 1 to 1 or more, this means one patient can have one or more different parts of clinical information about that patient's condition. The clinical information is then passed onto the Receptionist via the request card, who organises and prioritises the requests, which are then sent to the Doctor. The doctor checks and signs the requests if the examinations are to go ahead, these form a number of Authorised Requests, which are then sent to the Radiographer, who will perform the examination, the authorised requests may sometimes be checked by a Radiologist, this business worker has been indicated on the diagram by a generalised arrow from the Radiographer. The multiplicity between Requests and Authorised Requests is highlighted by a green association line, the multiplicity is 1 to zero or more, this is because a number of requests may all not get authorised therefore in the worst case this is zero or more.

3.3.3. Business Use Case Diagram

The business use case diagram is used to primarily identify the main elements and process that form the current system. The use of actors and processes, termed use cases, help to quickly establish from a high-level, process ownership.

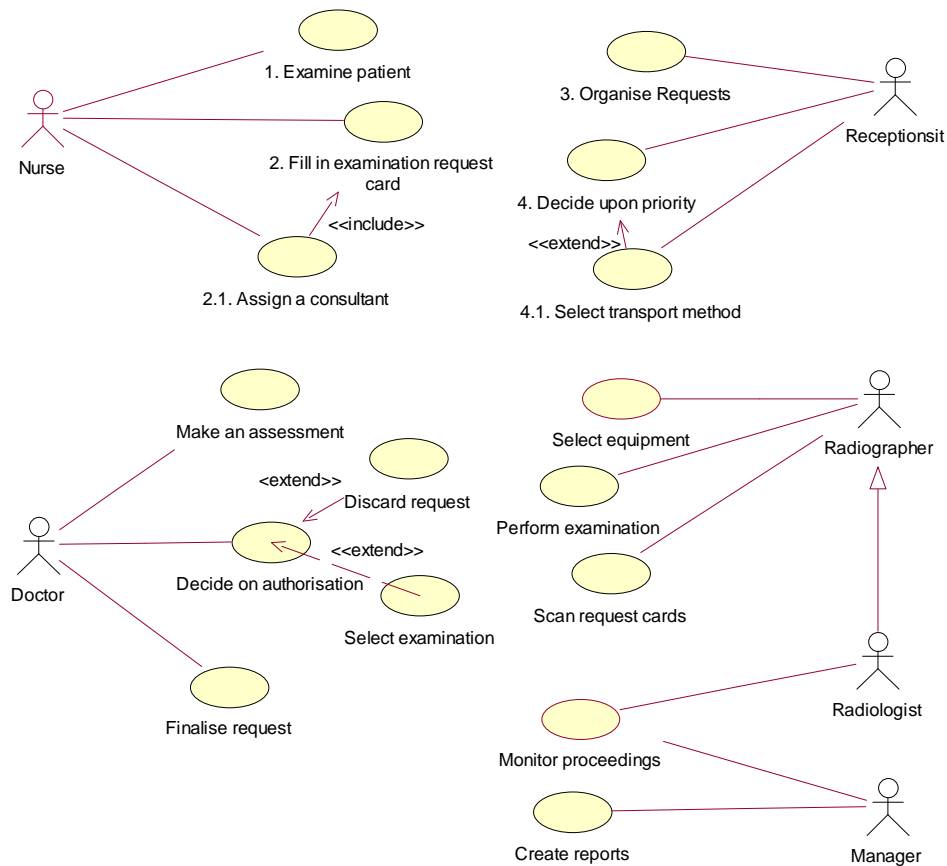


Figure 3.3: Business Use Case Diagram of department's manual processes

Figure 3.3 shows all the main duties and processes assigned to the business workers within the department; the processes and duties modelled relate directly to the current card-based system, of course the actors carry out other business activities. The 'include' text used here indicates that the process or duty is included in another use case, for example 'Assign a consultant' is included in 'filling in the examination request', not dissimilarly the term extend is also used, this relates to an expansion of a use case and can be thought of as a parent-child relationship. In the diagram 'Select Examination' and 'Discard Request' are both extensions of the Doctors authorisation decision, of which there are two possible outcomes.

3.3.4 Low Level Activity Diagram: showing a proposed electronic system.

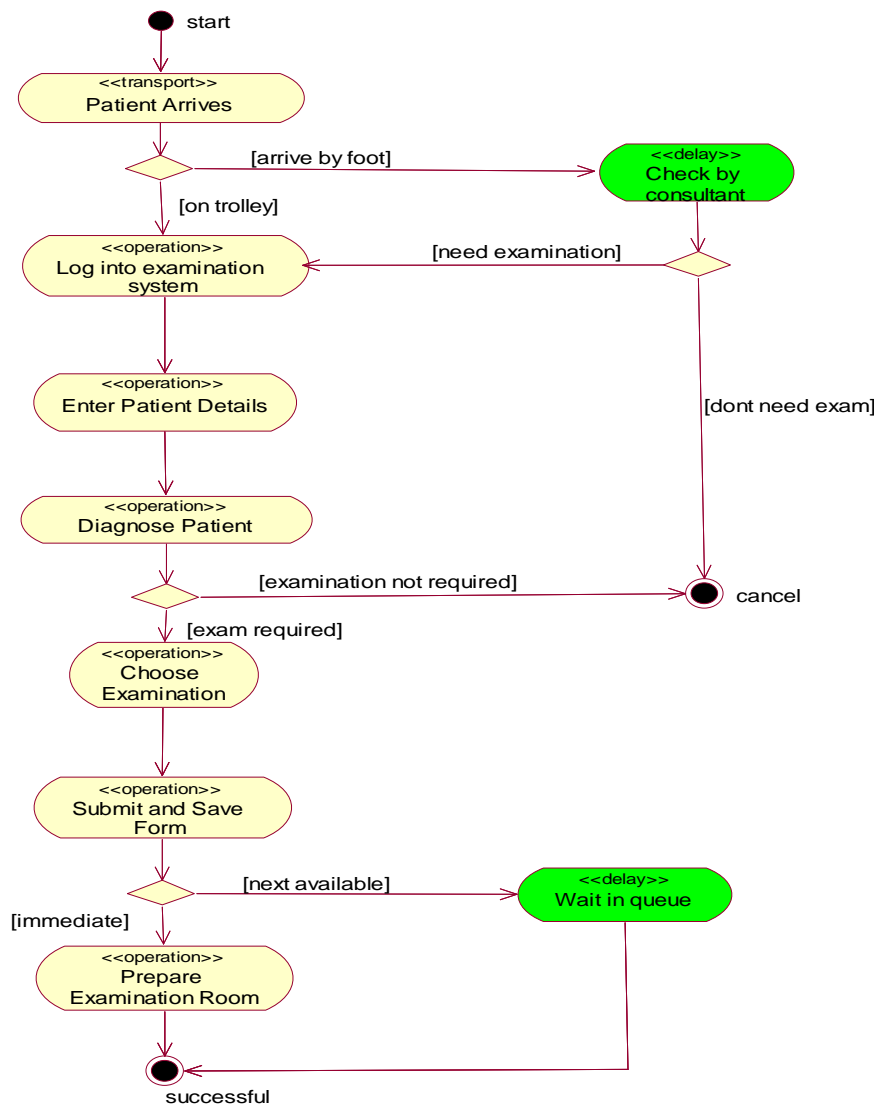


Figure 3.4: Low-Level Activity Diagram requesting an examination with an electronic system.

Figure 3.4, above, shows an activity diagram with an introduction of a new electronic system and the processes involved from a low level. The diagram is a more detailed version of activity diagram, using ASME standard, for classifying each activity type. Decisions undertaken are visibly marked using decision diamonds. It is clear that the electronic system removes the associated transport delays, which were experienced with the card-based system; however there are still two delays, marked in green, which cannot be helped, even with the introduction of new technology. A full process report, with comparisons drawn from the new system and the current card-based system, can be found in Appendix D.

Chapter 4

Design

4.1 Introduction

System design is an important stage in prototype development. It allows for the graphical user-interface, layout and requirements of the system to be fully scoped before the actual technical implementation is started. The design phase has the intention of making the system implementation as easy and as clear as is possible. This chapter details the design solution for the examination request system; considering in the first instance the techniques used to design the system then proposing a design for the interface and the overall structure.

4.2 Design Techniques

The techniques behind the design are very important factors to consider before carrying out the actual design phase. It can be done with or without user participation; obviously without user participation the phase can be carried out quicker, however with user participation it allows another chance for requirements to be underlined and confirmed thus a design will be produced which is right for the end users; furthermore, according to Norman et al (2004), since the goals of testing design prototypes are for the benefit of human users, most usability testing methods naturally involve tests on real users and require other humans to interact with the prototype. It has been decided to include some user participation in the design of the system, as this can play a key role in producing an effective design, details of this are in section 4.2.1 below.

4.2.1 Design Walkthrough

According to Rubin (1994), the purpose of a design walkthrough is to find potential usability problems within an interface by envisioning a user's path through an early concept of a prototype. A design

walkthrough will be scheduled once the design phase is complete; this is to educate the attendees on the system and subsystem design and specifications. Radiological departmental staff and eventual system end-users will look at the design. The walkthrough itself will be a presentation of the system requirements and form designs for the new system, in which the various screens and steps will be explained to the staff. The staff will then be allowed to ask questions during the walkthrough to clarify aspects, which may be unclear, and make suggestions; the walkthrough should last between 20-25 minutes. The walkthrough will be carried out and feedback appended at the end of the final report. The walkthrough is important as I am following the waterfall model; which does not allow the developer to go back i.e. to change requirements, therefore the walkthrough acts as a checkpoint before moving on with the implementation phase.

4.3 Graphical User Interface (GUI) Design.

The appearance and layout of the system is of paramount importance. Having a fully functional system is not enough, as the system may be deemed a failure without a well designed and planned interface. Figure 4.1 below shows a scanned examination request card, which is scanned and stored into the Picture Archive and Communication System (PACS). A solution would obviously remove the need for digital scanning and would ultimately replace the PACS system.

Department of Clinical Radiology			Name	
Examination Request			WNU546	
Address			Telephone	
Consultant			DOB	
Department Required	Referring Category	Urgency	Sex M <input type="checkbox"/> F <input type="checkbox"/>	PAS No/Unit No
LGI - Main Site	Ward	24 hrs/ASAP	<input type="checkbox"/> Walking	<input type="checkbox"/> Chair
LGI - Clarendon Wing	OP	1 Week	<input type="checkbox"/> Cot	<input type="checkbox"/> Drip
Chapel Allerton	GP	Routine	<input type="checkbox"/> Bed	<input type="checkbox"/> OT
Cookridge	Other Hosp		<input type="checkbox"/> Trolley	<input type="checkbox"/> On Ward
			Transport required Yes <input type="checkbox"/> No <input type="checkbox"/>	
Clinical Information			Appointment Date	
			am/pm	
Examinations Requested			Diabetic? Yes <input type="checkbox"/> No <input type="checkbox"/>	
Radiography			Previous Examinations	
Ultrasound			Yes <input type="checkbox"/> No <input type="checkbox"/>	
CT			Date	
MRI			Place	
Nuclear Medicine			LMP	
Doctor's Signature			Allergy to Contrast Medium	
Doctor's Name (Block Letters)			Yes <input type="checkbox"/> No <input type="checkbox"/>	
Date			Pregnancy	
Mandatory			Yes <input type="checkbox"/> No <input type="checkbox"/>	
			Should examination proceed if menstrual period overdue?	
			Yes <input type="checkbox"/> No <input type="checkbox"/> Not applicable <input type="checkbox"/>	

Figure 4.1: Current Examinations Request Card

Although no major concerns have been raised by either staff or management regarding the current layout and structure of the request card, minor changes will need to be made when implementing the interface, keeping relatively close to the current structure with which users are familiar with. However before implementing the system, I will need to consider usability issues, which will be covered in section 4.5 which will detail a framework to follow when designing the GUI. The mandatory changes that will have to be imposed from transition of the manual system to the computerised system are marked on the card.

4.3.1 GUI Components: Swing vs. SWT.

Developing Java components, namely menu-bars and drop-down boxes, is a very expensive and time-consuming exercise; therefore it is necessary to choose an appropriate GUI component technology. Swing, is one possibility, and is a built in GUI component technology for the Java platform. Swing is actually a descendant of AWT technology, provided with the early releases of Java. Eckstein et al (2002) states that in a sense, Swing replaces AWT.

SWT, the Standard Widget Toolkit, is a competing alternative GUI component technology that is part of the Eclipse project. There is a large and growing community that is leaning towards SWT over Swing for GUI programming in Java.

However, Swing is generally superior to SWT on its own, according to Feigenbaum (2006), the reasons for this are that Swing has the added advantages of being built into Java technology and furthermore is entirely portable and has an enhanced architecture; Swing is also beneficial for advanced graphical applications, SWT on the other hand is advantageous to developers creating a system for one specific platform as SWT has advantages in host compatibility which includes integration with a number of host features.

Having considered the two component architectures, it would seem more appropriate to use Swing, as I will be creating a system to work on a number of different platforms therefore, the remainder of this chapter will refer to certain tools that work with Swing. If a tool happens to support SWT as well I will mention that fact, if I am aware of it, however I have not tested the SWT capabilities of any of these tools.

One of the requirements for the system was to support the clinical information collected about a patient by allowing for as much information to be stored as possible. This can be made possible by utilising Swing components in Java, enabling users to scroll downwards if the text exceeds the box without restricting what the user enters. Similarly, the examination request comments data entry box requires similar expansion, so the user is not restricted. Furthermore, by introducing a clinical information button, similar to a help button on standard applications, it will allow for a clinical help function to be implemented. The concept behind the clinical help function is to use what has been written about a particular patient including all the symptoms and medical facts known about them, which will be entered in the clinical information box, and if a medical examination cannot be decided upon, based upon that information, then by clicking on the clinical help button it will bring up a separate interface which will be structured in such a manner which will allow a nurse or qualified member of staff to find those particular symptoms which will then present extra information which the user can ask the patient, there will be a number of related questions which when complete will provide a specific examination recommendation based upon the information learned. The use of this information could vastly speed up the examination requesting process by reducing the need for a doctor's opinion, and thus allowing the examination request to be immediately scheduled; this will be explored in more detail in the next

chapter. A possible implementation solution for this function could be to use the Swing component, JTree, which would present the information in a Tree list form. The addition of this function will obviously mean a change in business process, a full list of business process changes will be presented to the department management, along with justifications for the changes, including how it can save the department time and human resources; this feature will also be fully evaluated in the Evaluation chapter. The full process report can be viewed in Appendix D.

4.3.2 Help information

The use of help information can further increase the usability of the system, according to Nielsen (2003), the ultimate failure of a system is, “a failure to provide the information users are looking for.” Java has built in Swing components, which allow users to be able to create ‘tool tips’ which provide help to users of the GUI when hovering over a specific component. Creating a tool tip is relatively easy, using the *setToolTipText* method which sets up a tool tip for a specified component. To add tool tips to three buttons, only three lines of code is required; when the user of the program then pauses with the cursor over any of the programs buttons, the tool tip for that button appears. Figure 4.2 below shows a drawing of the tool tip that would appear when the cursor hovers over the left button.



Figure 4.2: Tool Tip Example - accredited to <http://java.sun.com/>

4.4 System Usability.

Usability is a very important aspect of any system design. There are usability issues to consider before designing the different screens. To help decide upon appropriate usability for the system, a framework for testing system usability was used, devised by Jacob Nielsen. Nielsen’s framework is known in human-computer interaction as heuristic evaluation, providing guidelines for producing a high quality, clean user-interface. These guidelines were applied to the existing card-based system and improvised to be able to ask myself how each part of the present interface could be improved for an electronic version. The improvised guidelines, outlined by Nielsen, J. (1994), are listed below; supplemented with how they can be directly applied to the creation of a new user interface.

- **Visibility of system status** – The system should always keep users informed of goings on through feedback; this can be achieved by implementing dialog boxes, error messages and other alerts.
- **User control and freedom** – Users should be able to undo mistakes they make; this can be implemented by adding edit buttons and allowing fields to be changed.
- **Consistency and standards** – Processes and layout should be consistent and familiar to the user. It is important the GUI follows platform conventions.

- **Error prevention** – It should be made as difficult as possible for error messages to occur. Range checks, type checks and other database checks will be imposed to ensure this.
- **Recognition rather than recall** – Users shouldn't need to use their memory, all the information should be presented before them. This can be achieved by not having excessive number of pages, boxes and short-cut keys – only what is required, organised in a logical fashion.
- **Flexibility and efficiency of use** – Users of different expertise should be catered for by the system; GUI should be created to suit both the novice and expert user.
- **Aesthetic and minimalist design** - Dialogues should not contain information, which is irrelevant or rarely needed. This can be imposed by only putting into the interface what is entirely necessary.
- **Help users recognise, diagnose, and recover from errors** – Error messages should be expressed in plain English. This will allow users to understand where they have gone wrong.
- **Help and documentation** – There should be some documentation on the system explaining how it works. This can be done at implementation time or after with user documentation.

4.4.1 Colour

Parker (1997) states that colour can have both positive and negative implications. If colour is used it can brighten up the graphical user interface and draw attention to important parts of the system. It can also be used in error messages and alerts to draw the user's immediate attention. However, Parker also warns of colour misuse. If used incorrectly, it can lead to confusion and disorder; it may be that users don't like to look at the interface for too long as a result. It is not wise to use too much colour and colours which clash can lead to problems, in particular with text. Black text on a white background reads better than white text on a black background say.

4.4.2 Form Designs

Three different data entry forms have been designed:

- (1) The login screen – which offers security against unauthorised personnel.
- (2) The main patient and examination data screen – includes all the clinical information about the patient and the examination required.
- (3) A doctor confirmation screen – confirms the examination or rejects it if it isn't to go ahead, this is only required if the nurse or qualified person was unable to decide on the examination.

Each form has been designed to allow for easy navigation, the data entry form for patient details now reads from left to right, instead of right to left like the old card-based system. The login screen will include the NHS Trust logo, removing it from the examination request form to allow for more space. It consists of two text boxes, one for a member of staff and authorised system user and the other for their password. The password entry will obviously show as a series of stars or hashes for added security.

required, this screen design can be seen in figure 4.5 below. Also introduced are scroll bars and boolean true/false entry boxes, depicted by the small square boxes on the screen.

The interface has been structured in a way which a user would carry out the tasks; this is so that a user is not going up and down the interface and can start from the top left and progress until at the bottom right of the GUI.

The doctor confirmation section resides in the bottom of the interface and allows for a doctor to search a patient who exists in the database and confirm an examination for that patient. This section allows the doctor name to be entered, and his unique id number, the date to schedule the examination and any concluding notes.

Clinical Support System	
<input type="checkbox"/>	Begin Clinical Questions
<input type="checkbox"/>	Does the patient have leg-pain?
<input type="checkbox"/>	Does the patient also have back pain?
<input type="checkbox"/>	Does the patient have lower back pain?
<input type="checkbox"/>	Does the patient have upper back pain?
<input type="checkbox"/>	Has pain in lower back persisted for 3-6 months?
<input type="checkbox"/>	Does the patient also have bowel incontinence?
<input type="checkbox"/>	[Possible CT scan required]

Current Selection: Patient Also Has Bowel Incontinence
--

Figure 4.5: Clinical Support Screen.

4.5 Database Design

As all patient data is currently held in a Microsoft SQL Server database within the department, the interface will need to provide cross-platform java database connectivity (JDBC); this will allow the Java examination request system GUI to connect to the SQL Server database. Furthermore, because the system is new and will now rely on existing technology, a new database schema will need to be set up. This section will detail the entity-relationship modelling and the table normalisation. The database should be designed in a way that tasks are performed correctly, fast and efficiently. Data should be stored only once and there must be a correct relationship between tables, the design should help in achieving this.

4.6 Entity Relationship Modelling

This section details the different relationships that exist between the different entities. One doctor can enter many examination details. A patient_id will be unique to an individual doctor, two doctors may not enter the same patient's exam details therefore this is a one-to-many relationship.

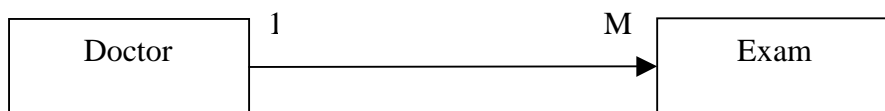


Figure 4.6.1: ER diagram for Doctor and the Exam they confirm.

One consultant can be assigned to many patients. A patient will be unique to a Consultant, a patient may not be assigned to many consultants – therefore the relationship is one-to-many.



Figure 4.6.2: ER diagram for Consultant and the Patient they are assigned to.

One patient can have many different examinations with each different examination over time assigned different exam_id's, therefore the relationship between a patient and an Exam is a one-to-many relationship.

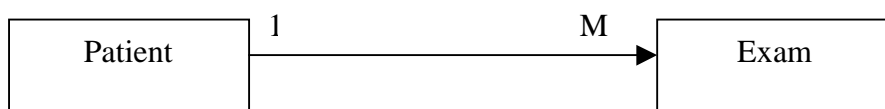


Figure 4.6.3: ER diagram for Patient and the Exam they have.

One consultant can be assigned to many different examinations; therefore the relationship is one-to-many.

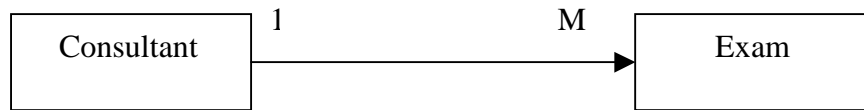


Figure 4.6.4: ER diagram for Consultant and the Exams they are assigned to.

4.7 Table Design:

Doctor Table			
<i>Column Name</i>	<i>Description</i>	<i>Data Type</i>	<i>Check Constraint</i>
Doctor_id	Unique doctor number	Integer	
Doctor_name	Doctors full name	varchar(30)	

Figure 4.7.1: Doctor Table design.

Examination Table			
<i>Column Name</i>	<i>Description</i>	<i>Data Type</i>	<i>Check Constraint</i>
Exam_id	Examination unique id	Integer	
Dept_req	Department required	varchar(20)	
Ref_category	Reference category	varchar(16)	
Urgency	Urgency of exam	varchar(12)	
Status	How patient arrived	varchar(10)	
Appt_date	Date of appointment	Datetime	
Clininal_info	Clinical Information	varchar(100)	
Diabetic_info	Is patient diabetic?	Boolean	Entry 'Y' or 'N'
Exam_required	Examination required	varchar(15)	
Date_of_exam	Date exam should be	Datetime	
Extra_info	Any extra information	varchar(80)	
Allergies	Allergy to medium?	Boolean	Check entry is either 'Y' or 'N'
Pregnancy	Is patient pregnant?	Boolean	Check entry is either 'Y' or 'N'
Menstrual	Should exam proceed if menstrual o/due?	Boolean	If pregnancy = 'false' menstrual should be 'N/A'
Doctor_id	Doctors unique id	Integer	
Patient_id	Patients unique id	Integer	

Figure 4.7.2: Examination Details Table Design.

Patient Table			
<i>Column Name</i>	<i>Description</i>	<i>Data Type</i>	<i>Check Constraint</i>
Patient_id	Unique patient id	Integer	
Patient_name	Patients full name	varchar(30)	
Patient_addresone	Patients address	varchar(25)	
Patient_addresstwo	Patients address 2	varchar (25)	
Patient_telno1	Landline Number	Nchar(12)	
Patient_telno2	Mobile Number	Nchar(12)	
Patient_dob	Date of birth	Datetime	
Patient_sex	Sex of patient	Boolean	Check entry is either “M” or “F”
Patient_transport	Any transport req?	Char(10)	
Consultant_id	Id of assigned consultant	Integer	

Figure 4.7.3: Patient Table Design.

Consultant Table			
<i>Column Name</i>	<i>Description</i>	<i>Data Type</i>	<i>Check Constraint</i>
Consultant_id	Unique consultant id	Integer	
Consultant_name	Name of consultant	varchar(30)	

Figure 4.7.4: Consultant Table Design.

4.8 Complete ER diagram:

A complete Entity Relationship diagram can be found in Appendix C.

Chapter 5

Implementation

5.1 Introduction

This chapter details the implementation of the system prototype; firstly the technical platform for the prototype is discussed and the advantages explained along with the actual implementation of the system GUI, and related components. The connection to the database and the queries which were used are then described, showing how the database was linked to the GUI to enable data to be stored and queried

5.2 The Technical Platform

The programming language which was chosen for the creation of the system prototype is Java, a high-level programming language developed by Sun Microsystems. Java is an object-oriented language, similar to C++, there were many underlying reasons for choosing Java as a platform for the creation of my system, in particular the object-oriented nature was a huge benefit as it supports code-reuse, this means that components did not have to be coded from scratch and could be re-used within my application, under stringent time-constraints this was a huge benefit to me. Furthermore, compiled Java code can run on nearly all computers, because Java interpreters and run-time environments named Virtual Machines, exist for most operating systems, therefore I knew that my system would work on the hospitals departmental computers. The DBMS which was chosen to store and manage the patient, doctor and examination data was Microsoft SQLServer, one of the primary reasons for choosing the DBMS is because the hospital currently have this in place across departments and it is used across the hospital's network. If I were to choose a different DBMS such as Postgres, either the problem owner or I would need to migrate the database to be read by SQL Server; this would be both a time consuming and expensive exercise. SQL Server however was an automatic choice from personal preference; I was both experienced and comfortable in its use and was able to set-up tables by running SQL and set constraints

by creating database “triggers”, this will all be detailed later in this chapter.

5.3 GUI Implementation

The GUI was coded within Eclipse, an open source development environment providing an extensible development platform and application framework for building software. The benefit of using Eclipse is that code can be created and any errors within the code are displayed in real-time next to the relevant line of code, opposed to at run-time, therefore errors can be detected and rectified before compiling the program. A simple login class was created first which takes a unique user-id and password and logs authorised users into the main system. For testing purposes, the username and password were hard-coded with the default set to “sean” for the username, line 3, and “sean” for the password, line 4, the code below shows some of the parameters passed resulting in a successful login if the username and password are correct.

```
1. String strUserName = request.getParameter("userid"); // the user id
2. String strPassword = request.getParameter("userpassword"); // the password
3.    if(!(strUserName == null) && strUserName.equals("sean"))
4.        && !(strPassword == null) && strPassword.equals("sean")) {
5.        //valid user create the session and give proper msg
6.        strResult = "Login successful!!!!";
```

The Examination Request class was created secondly; this was the main interface for the system and formed the backbone to the system. The interface made extensive use of Java Swing components, including JTextFields for being able to enter information such as patient name and address, JComboBoxes which act as pull-down menu’s, this meant that system users do not need to key in recurring information such as the type of medical examination, it can be simply selected from a drop-down list. JButton components were also used to click on and call other parts of the system such as the “Save” and “Search” buttons which when activated, query the database. Making use of another swing component, JPanel, held the whole interface together. Different parts of the system were layed out in JPanels, this made it easier to present the information to the end-user. One important part of the GUI implementation was being able to capture the users actions. This was made possible in Java by the use of ActionListeners. ActionListeners are probably the easiest, and most common, event handlers to implement, they were used primarily to confirm a user’s selection, however they were also used on all the buttons on the GUI screen to call other classes or to execute certain commands. The code below shows the implementation of an ActionListener on the GUI’s “Search” button.

```
1. search_button.setText("Search");
2.    search_button.addActionListener(new ActionListener() { // the code to set new A-L
3.        public void actionPerformed(ActionEvent e)
4.        {
```

```

5.         String findSQL = "SELECT patient_id from patient_tbl WHERE "
6.             + "patient_name='" + patient_name.getText() + "' "
7.             + "AND patient_addone='" + add_lineone.getText() + "' "
8.             + "AND patient_addtwo'" + add_linetwo.getText() + "'";
9.
10.        SQLExecutor se = new SQLExecutor(findSQL);
11.        JTable rt = se.getResults();
12.    }

```

From the code it is easy to see how the ActionListener responds to an action, in this case it is the action of clicking on the “Search” button. Line 2 shows how a new action listener is added to the search button, Line 3 then initialises the ActionListener and declares that the following commands 5-11 be carried out if an action is performed. The command that is actually carried out is an SQL Select query which finds the relevant table attributes in the database, runs the SQLExecutor class, which interprets the SQL command then displays the results in a JTable; the SQL which is performed will be detailed later in this chapter. The “Save” button similarly functions the same as the “Search” button, adding a new ActionListener to detect an event then performs a command, which instead of searching for information, collates and adds all the details entered on the form to the database.

5.4 Clinical Support Implementation

The clinical help support screen was implemented as a separate class to the main examination request system. An action listener was implemented in the main GUI class that called this on the user clicking the “clinical help” button. The clinical help is an important addition to the examination request system; it assists in making a decision on the medical examination required for the given clinical information known about the patient. The purpose of the help support is to save time and resources by making quicker decisions without requiring the confirmation from a doctor; the system however does rely on specific symptoms to be known about the patient before a suggestion for the type of examination is provided. By building up a portfolio of medical information related to the typical symptoms, which may lead to a particular medical examination, I was able to program and code this information into Java for fast and efficient retrieval. There were a number of ways in which I could have utilised Java tools and components to display this information; however I decided that by using an expandable JTree the information could be quickly searched and appropriately displayed in a linked form, so the path to the final conclusion on the examination could be visible. The key to the JTree component is that selections are linked, this was appropriate for linking the symptoms together and following the correct path.

According to Gosling et al (2003) the simplest and most common way to use JTree is to create objects of type DefaultMutableTreeNode to act as the nodes of the tree, nodes that have no children will be displayed as leaves.

The code below shows an extract from the code detailing how the clinical information was implemented and linked together using the parent, child and grandchild node model.

```

1.  for(int grandParentIndex=1; grandParentIndex < 2; grandParentIndex++) {
2.      grandParent = new DefaultMutableTreeNode("[Does Patient Have Back Pains?] " +
3.          grandParentIndex);
4.      root.add(grandParent);
5.
6.      for(int parentIndex =1; parentIndex < 3; parentIndex++) {
7.          parent = new DefaultMutableTreeNode("[Does the patient also have leg pain?] " + parentIndex
8.          + "." + parentIndex);
9.          grandParent.add(parent);  }

```

From the code it is clear to see how using a JTree can link a portfolio of medical information. The information gets displayed as a series of folder icons which ask specific questions, if the answer to each question, like on line 2, is “Yes” then the user progresses by expanding out that folder and then answering the next question in the next node, line 7. This sequence continues until enough information is known and a suggestion can be made to the user. The screen shots of the different permutations of this system can be found in Appendix I, and its usage fully explained in the User Guide, Appendix F.

5.5 Setting up the database

The database needed to be created to allow a connection from the medical examination request system. The database was set-up by writing a series of ‘CREATE TABLE’ commands in SQL, as tables are the basic structure where data is stored in the database. In creating the database table all the attributes were declared and their accepted values. Check constraints were also included to ensure database integrity, set to reject any incorrect data entry; the primary and foreign keys were also set. The code below shows the SQL code for creating the patient table in Microsoft SQL Server.

```

CREATE TABLE patient_tbl (patient_id integer primary key,
    patient_name varchar(40) NOT NULL,
    patient_addone varchar(25) NOT NULL,
    patient_addtwo varchar(25) NOT NULL,
    patient_telno varchar(12) NOT NULL,
    CONSTRAINT check_tel CHECK (patient_telno LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    patient_dob datetime NOT NULL,
    patient_sex varchar(1) NOT NULL,
    CONSTRAINT check_sex CHECK (patient_sex = 'M' OR patient_sex = 'F'),
    patient_transport varchar(10) NOT NULL,

```

consultant_id integer NOT NULL references dbo.consultant_tbl(consultant_id));

The patient_id was set as the table's primary key, accepting any unique integer value. The telephone number includes a check constraint to check that the number is in the standard number format of 11 consecutive numbers with no spaces, if this is violated an error will appear and the insert of a row in the table will not be allowed until it has been changed to the correct format. Similarly, a check is performed on the patient's sex, ensuring that entry is either 'M' or 'F'. Finally, the consultant_id is set as the table's foreign key, this means that it references a primary key from another table; in this instance it is referencing the Consultant ID from the Consultant table. All attributes are set to reject any 'null' or blank entries.

5.6 Connecting to the Database

My database was set-up on the University of Leeds csms11 server. To enable a correct connection a unique username and password was created by the School of Computing technical support team, the username and password was subsequently assigned to my SQLServer Database on the csms11 server, under "users", this was the mandatory and standard way of ensuring that my database could be recognised by the Java Database Connectivity (JDBC) drivers.

"The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases." Van Haecke (1997).

The JDBC API made it possible to do three things which will be covered in this chapter:

1. Establish a connection with the database.
2. Send SQL statements
3. Process the results

The code to connect to the database is listed below, the String URL being read by Java denotes the database platform attempting to connect to, in my case this was Microsoft SQLServer, the string also takes the local host and opens the default port of 1433, shown on line 1, this then allowed my database fyp_scs3sw to be located. The password has been removed from the code listing below for security reasons.

1. *private String url = ("jdbc:jtds:sqlserver://csms11.leeds.ac.uk:1433/fyp_scs3sw");*
2. *private String username = "scs3sw";*
3. *private String password = "";*
4. *connection = DriverManager.getConnection(url, username, password);*
5. *System.out.println("Database connected");*

5.6.1 Driver Importance.

It was of paramount importance that a driver was chosen to allow successful connection to the database from my Java application. The JDBC API defines the Java interfaces and classes which are used to connect to the database and send queries; whereas the JDBC driver actually implements these interfaces and classes for the DBMS vendor. There are different kinds of driver which can be used; a Java type 4 driver was selected for my application, also known as a native-protocol driver, converting JDBC calls directly into the vendor-specific database protocol.

“The driver is written completely in Java and it provides better performance over Type 1 and Type 2 drivers as it does not have the overhead of conversion of calls into Open Database Connectivity (ODBC) or database API calls.” Reece (2000).

```
1. private static String database = ("net.sourceforge.jtds.jdbc.Driver");
2. private boolean tryConnect() {
3.     try {
4.         Class.forName(database);
5.     } catch (ClassNotFoundException e){
6.         System.err.println("Not working");
7.     }
```

The code above shows how the driver is set as a String named database, line 1, then passed by the ‘tryConnect’ method, line 2, which attempts to locate the driver and connect, line 3, if the driver isn’t found then an exception is caught and an error is returned, line 6.

5.7 Writing and Executing the SQL

Because data types in SQL and Java are not the same, there has to be a mechanism for transferring data between an application using Java types and a database using SQL types. The JDBC API documentation was my main source of reference for creating and modifying SQL commands as it contained a detailed, yet simple to follow range of methods to follow. According to Fisher et al (2003) The Java API provides three sets of methods, the `ResultSet` class retrieves SQL `SELECT` results as Java types, `PreparedStatement` class sends Java types as SQL statement parameters and finally `CallableStatement` class for retrieving SQL `OUT` parameters as Java types. All three methods were used effectively in a class named `SQLExecutor.java` to execute SQL based on the “SELECT” and “INSERT” commands.

```
1. public void executeSQL(String sql) {
2.     if (tryConnect()) {
3.         try {
4.             Statement statement = connection.createStatement();
5.             System.out.println("Executing SQL: " + sql);
```



```

6.          // now look to see if we expect results from the DB
7.          if (sql.startsWith("SELECT")) {
8.              //execute the query
9.              statement.executeQuery(sql);
10.             //get the result set
11.             ResultSet r = statement.getResultSet();
12.             //change to an ArrayList for easier manipulation
13.             results = resultSetToJTable(r);
14.         } else if (sql.startsWith("INSERT")) {
15.             //execute the query
16.             statement.executeUpdate(sql);
17.         }

```

The commented code above helps to explain how there is a mechanism in place for transferring data between my Java Interface and the database using SQL types. The code shows that where the SQL starts with “SELECT” it will execute and get the results and put it in a ResultSet, the ResultSet is then changed to an ArrayList for easier manipulation in Java. Similarly, if the SQL starts with “INSERT” this will execute the executeUpdate(sql) statement, this would be called when inserting information into the database.

5.8 Processing the results

The final stage in the transfer of database to the Java application is being able to process the results. There are different ways in which the SQL ResultSet can be displayed, I implemented this stage by turning the results of the ResultSet into an ArrayList and creating a JTable based on this; I did this because the ArrayList in Java allows for data to be collected and stored effortlessly in a list form and the JTable allows for simple population and displays the information in a legible and structured form. The method to process the results of the queries was implemented in the same SQLExecutor class as the connection and SQL statements.

```

1. public JTable resultSetToJTable(ResultSet r) {
2.     JTable temp = null;
3.     Vector columnNames = new Vector(); //get the column names, store them in a Vector
4.         for (int i = 1; i <= noOfCols; i++) {
5.             String name = rsmd.getColumnName(i);
6.             columnNames.add(name);
7.         }
8.     Vector rowData = new Vector(); // make a Vector of Vectors for the rowData
9.         while (r.next()) { // cycle through all the rows in the ResultSet
10.             Vector currentRow = new Vector();

```

```

11.                                     //cycle through the columns in the current row and get the data
12.                                     for (int i = 1; i <= noOfCols; i++) {
13.                                         currentRow.add(r.getString(i));
14.                                     }
15.
16.     rowData.add(currentRow); //add the Vector of strings for current Row to the rowData
17. }
18.     temp = new JTable(rowData, columnNames); //now Create a JTable based on this

```

Above is an extract of some of the most important procedures in transferring the ResultSet to a JTable in Java. Firstly the database column names are read and stored into a vector, line 3. A vector of vectors was then created for the rowData, line 8, and the same for columnNames, line 3, all the columns and rows were cycled through and the data was added into the vectors. The rowData and columnNames vectors were then added into a JTable by assigning them to the JTable “temp”. A new method was then coded to call the result method and get the results from the JTable, the database then could be successfully queried and data could be displayed in a JTable in my Java application.

5.9 Problems Encountered

The implementation of the interface proved to hold no significant problems, however one problem encountered was positioning all the components correctly on the GUI. Java makes use of containers and when using a lot of containers, it can be difficult to know which is which and therefore components can get added to the wrong container, also numeric attributes are used to position text on the screen, this process of positioning the components and text was a very tedious and time-consuming exercise. I also experienced some minor database connectivity problems, with the database not detecting the drivers to connect to the database; ensuring the driver files were in my workspace and changing the class-path file to point to that particular location easily solved this.

5.10 Screen Sources

The system prototype is a project deliverable; therefore a full variety of screen shots have been taken and uploaded to the Internet. The links can be viewed in Appendix I, found at the end of the report.

Chapter 6

Testing

6.1 The need for testing.

According to Emprend (2005) testing is a crucial part of the product development process; it is used to ensure that for a set of specified inputs the design produces the accurate and expected output results. Testing is also a procedure which ensures that a product has met its specified requirements, and in a state that it can be used both efficiently and effectively by the system end users.

6.2 Types of testing.

Almstrum (1999) describes three different types of testing methods. One testing method being black box testing, this method disregards anything which is internal such as the system code and instead focuses on the actual functional system performance. The second testing method is the reverse of black-box, titled white box testing; this focuses on the internal aspects like the system code. Walkthrough testing is the third testing method; this is completed by a system user to test the system front-end, finding any faults or errors in the system. All these types of testing methods are appropriate for testing the examination request system to ensure the code is correct, the functionality is satisfactory and the system is usable.

6.3 Code testing.

The approach taken to test the code has been automated and continuous, to automate testing a testing framework was required, I used an open-source tool provided within eclipse, called JUnit, it was beneficial for several reasons, I did not have to write my own framework, it is open source and

therefore I did not have to buy the framework, there were a lot of examples from other developers in the open-source community so I was able to quickly understand the layout and style and finally it allowed me to separate my test code from my product code whilst also allowing for easy integration into my build process. The testing was very much incremental and enabled me to verify my work method by method, once I created a method I could then run a JUnit test on that method and this would indicate whether the code was correct or incorrect, if incorrect JUnit would list reasons why the test may have failed, leading to quicker resolutions to any problems. I experienced test fails frequently throughout the coding phase of the project, which was fully expected, and JUnit helped in quick resolution to the problems, not only correcting the problem but improving the code by suggesting different ways of fixing the problem which I would not have thought of. This form of regression testing has the advantage of not requiring any extra code testing once the program is complete, as it has been tested step by step to the very end of the implementation. A sample of some of the JUnit test coding can be found in Appendix E at the end of the report.

6.4 System Testing.

The testing of the complete system was achieved using different types of input, recorded was the expected output and then the actual output. Included in the system tests were range checks, as well as the actual data types, the database was tested as a result to ensure the correct data entered into the system GUI appeared in the correct columns, in the correct tables in the database. As well as checking the system and database functionality, I also tested for errors by entering erroneous data such as duplicate records to check that it could not be appended into a database table. The final stage in the system test was to ensure that the system was tested against the user requirements to guarantee the objectives were met. Once satisfied that the system was fully tested and functional I opted for user interaction, the aim of this was to get alternative input from a different variety of users, this was accomplished by getting users to walkthrough the system. After the users had finished they were interviewed and their responses recorded, this ensured that any faults could be highlighted that may have been previously overlooked and furthermore allowed any usability issues to be raised, which would then allow for the issues to be recorded, as known, and placed in the possible system changes section, in the user guide.

6.5 Functionality Testing.

The figure below is an example of the testing procedure performed on every aspect of the system GUI, including all the input boxes and action buttons, to ensure the correct result is gained. The figure below is an extract from the full testing table, the testing table in full can be found in Appendix E at the end of the report. The figure shows how different areas were focused on and the different permutations tested for an expected result, if an actual result was different to the expected

result then this allowed a problem to be identified, it could then be resolved and reported on.

Tested Area	Input	Expected Result	Actual	DB Check	Problem Resolved
<i>Exam System:</i> Testing exam data goes into database Exam Table	Valid Exam Details	Inserts exam details into the Exam Table	<i>Confirmation Message:</i> “Data has been added into the database”	Data has been added correctly	N-A
<i>Exam System:</i> Testing the search button returns an existing patient stored in the DB.	Valid Patient ID number	Patient details and related examination details are returned to the system.	Patient Returned and rows displayed.	Patient returned matches the patient stored	N-A
<i>Exam System:</i> Testing the search button doesn’t return a patient not in the database	Invalid Patient ID number	Error Exception Thrown – patient is not in the database	Error Exception – Patient ID does not exist	The patient ID is not in the database	N-A
<i>Exam System:</i> Testing the phone number field with an incorrect format	A letter	When attempting to add to the database will return with an error	Violation Error – cannot add to the patient table	Was not added to the patient table	N-A

Figure 6.1: Data Testing Extract.

Overall, the testing was successful as it confirmed that the GUI was working correctly by producing errors or by correctly inserting and updating tables in the database, the testing also uncovered two problems which may not have been discovered had a rigorous test not been performed, after entering a patient id and a search performed to retrieve the data from the corresponding tables in the database, the data would fill the appropriate fields, however the patient_id would disappear. This was not necessarily an error in the coding, but merely an omission to the SQL Select code, whereby the patient information was retrieved where the patient_id matches a patient_id in the database, however I forgot to include a line of code to prevent the patient_id field from being cleared. There were two possible solutions to this problem, both included adding one line of code, I could either prevent the id field from being cleared or over write the id field with the one from the database. I

chose to prevent the field from being cleared as I saw it pointless overwriting the same value from the corresponding number in the database. The second error was regarding the doctor's exam confirmation check box, when the GUI loads the confirmation box is always checked. However, a doctor who wants to confirm a medical examination should only check it. By testing this in the eclipse development environment I could now see from the report console that this was actually trying to update a table within the database, the following print statements were displayed:

```
Running executeSQL(UPDATE exam_tbl WHERE (exam_requested,exam_date,doctors_notes)  
VALUES ('Examination Confirmed','');)
```

```
Initiating connection...
```

The potential problems this may cause is that an examination may be inaccurately marked as confirmed by a doctor, therefore any doctor receiving an e-mail to confirm a patient's possible medical examination will see that it has already been completed, obviously this can have serious consequences and would also cause time delays. The solution to this was very simple, all that was required was to initialise the doctor confirmation and set it to a NULL state, this resulted in the box not being checked when the GUI was loaded up.

6.6 Meeting User Requirements.

As this system was built to prototype standard it was not expected to reach all the user's requirements, only one iteration of the system build and test has been performed, obviously this allows for further future expansion and testing. The first iteration, however, has focused on capturing all the required data for a qualified member of medical staff to be able to request a medical examination for a patient and furthermore allowing a doctor to confirm any examinations, which could not be decided upon. The prototype also includes a clinical support tool to assist staff in making their decision, which has been tested in terms of its functionality and accuracy. Testing the system against the user requirements has enabled problems to be identified in the first instance and allowing for them to be corrected accordingly.

6.7 User Walkthrough.

It was decided to include some user interaction in the testing of the system. The system prototype was provided to two friends who study Computing within Leeds University to allow them to walk through the system and check for any errors; they were also shown the card-based system before hand. The user walkthrough results have been summarised, the general consensus was that the GUI was very well constructed and similar in style to that of the card-based system; therefore it was easy to distinguish what everything was from the old system. The background colour was deemed

appropriate and made the text and the components easy to read and discern.

6.7.1 Log-In Screen.

All users agreed that the log-in screen was very standard, similar to that of a website log-in or a normal password protected program, the recognisable appearance of the screen in effect made it easy to use. There were no criticisms or possible enhancements which arised from the use of this screen.

6.7.2 Examination Request GUI.

One of the first comments made by the users was that the size of the interface screen was able to be changed, which made the components spaced out which in effect made the interface harder to use, both users agreed that it would be better if a user was not able to change the size of the interface as it is not really necessary. It was suggested that to make the format of the date more understandable to users, it could be set to a default value and cleared once a user clicks on the field; this was again agreed by both users and considered a possible improvement. Another comment was that the clinical information box allowed too much information to scroll off the screen horizontally, requiring users to press return at the end of each line to break up the text, an improvement would be to add code to ensure information can be typed without the user always needing to press return. There were no further comments on this part of the system, it was concluded that the design and coding appeared to be done very well.

6.7.3 Clinical Support Screen.

The first impression from both users was that the clinical support screen looked unnecessarily large, however after discovering that the packages expanded to reveal further questions, the users were impressed. Both users felt they were not in a position to give an expert opinion on whether there were enough questions presented, therefore it was concluded to be acceptable. However, one particular problem occurred, when the user closed the clinical information screen by clicking the cross in the top right, it closed down the main interface too, it was decided that upon closure only the clinical information screen should close; this was noted as a possible change to make.

Chapter 7

Evaluation

7.1 Introduction.

The evaluation concludes the final stage of the waterfall methodology. Evaluating allows for the produced solution to be assessed, according to Dix et al (2003), the role of evaluation is to “*test a system to ensure it behaves as expected and meets the user requirements.*” Evaluation has been carried out throughout the design phase using a design walkthrough and during the implementation phase by testing the system functionality; this allowed for any problems to be identified and corrected before progressing onto the next phase in the methodology. The iterative nature of the waterfall model makes it imperative that requirements are satisfied before moving on, and under tight time-constraints it is impracticable to go back to correct a phase, this made continuous evaluation of paramount importance. This chapter looks at the various evaluation methods that could be used to evaluate my system.

7.2 Information Gathering and Evaluative Methods.

The process of gathering feedback and information has been a common premise throughout this project and comes into full effect during the evaluation of the Medical Examination Request System. Jackson (2002) outlines a number of factors that need to be measured when deciding upon an information gathering method; one factor of particular importance is that the method used depends on the type of evaluation being performed. This evaluation is primarily concerned with evaluating the system in terms of its usability and improved quality of data and information, and as Preece et al. (2002) states, designers of software should not presume that by following the design guidelines, the system is ensured of good usability. Evaluative results can be used in a number of

different ways depending on the nature of the project undertaken, for example the results can be used to form a list of possible improvements to prototype models or future builds or iterations of the system may include the suggested improvements. As my evaluation will be performed on a completed system, the last statement is correct. Preece et al. (2002) also details a framework to guide evaluation, called DECIDE. The evaluation contains a checklist of tasks in helping generate a well-planned evaluation, clearly determining the goals. The framework is very extensive; however it can be adapted to evaluate my system; by adapting the framework I would be determining the evaluation objectives, choosing the evaluation paradigm and techniques, identifying the practical issues that need to be addressed and evaluating, interpreting and presenting the data; this framework, although not as extensive as the full DECIDE framework outlined by Preece, it provides a detailed and very easy to follow approach to evaluating my system.

7.3 Determining Evaluation Objectives.

Before performing an evaluation, objectives need to be determined to help accomplish three things 1) To bring focus to the purpose of the evaluation, 2) To describe the results I would like to achieve throughout the evaluation and 3) To describe the manner in which these results will be achieved.

The identified objectives are listed below:

1. Determine whether the system is fully usable to a range of different system users.
2. Identify whether or not the system meets the design requirements and incorporates and meets all the specified general requirements.
3. Decide whether or not the system is superior to the manual examination requesting system.
4. Determine if the user-guide aided staff in correct system usage.
5. Decide whether there could be any future improvements made to the system.

7.4 Choosing the Evaluation Paradigm.

Preece et al. (2002) identifies four types of evaluation paradigm, three in particular would appear relevant in relation to my Examination Request System, the first being 'usability testing' this is providing users with a set of formal tasks and measuring their satisfaction and performance, this would allow me to understand how good the actual usability and quality of information is for system users, usability would test the interfaces ease of use, data and information quality and performance. The second relevant evaluation method listed was predictive evaluation, which includes both a heuristic and plurastic evaluation. According to Nielsen (1994) heuristic evaluation assesses the ability of the user interface, guided, by a set of usability principles, known as heuristics. This approach may be appropriate because it is a relatively quick and inexpensive form of determining usability and evaluating the system. A plurastic evaluation may also be relevant, as the method involves some of the wider stakeholders such as the users, system developer(s) and

experts working together step-by-step to discuss the system usability issues. Although this method allows for many issues to be uncovered, it is a relatively slow form of evaluation, as many discussions need to take place for each individual task; furthermore all the stakeholders need to be present for the evaluation to be successful. The third evaluation method which may be appropriate to evaluating my system is 'Field Study' method outlined by Preece et al. (2002), the paradigm involves observing and interviewing the users of the system in their natural surroundings, this could be effective way of collecting information about user's reactions and impressions of the system and evaluating the results. However, the only downside to this approach is that the user may feel pressured by being watched, and may feel inclined to rush through their evaluation of the system. It has been decided on the above collection of different evaluation methods, that the appropriate methods for evaluating my system will be to evaluate the system Usability and also the use of heuristic evaluation. Usability testing will have the benefit that typical system users will be directly involved, therefore it is expected that there will be some qualitative and quantitative data returned. I have decided to use two methods of evaluation opposed to just one, I feel Usability testing on it's own will not fully allow me to evaluate the quality of information, as according to Robson (2002), users sometimes say what the tester wants to hear rather than what they truly think; heuristic evaluation will allow me to add further weight to the evaluation and will confirm, by the use of experts, the strengths and weaknesses of the system.

7.5 Usability Testing.

Through involving users directly involved with the day-to-day use of the previous system will yield more accurate results opposed to users who have no previous knowledge of the previous system as it allows the usability to be compared with the usability of the previous system, this will allow the user to gauge the actual quality of the data and information quality. However, non-previous system users can also test the system; this will give a mixture of participants and will allow me to gauge how competent the user is with the new system without any previous knowledge. Usability testing is performed, according to Preece et al. (2002) by the evaluator setting stringent tasks that the user must work-through and complete. The result output is likely to centre on particular areas where users make unforeseen mistakes or where tasks are performed in an unexpected manner.

7.5.1 Test Users.

Nielsen (2000) states that in order for a usability to be more successful, it should be aimed at the target user group. My participants will be made up of doctors, nurses and qualified medical staff who will be eventual users of the system. As the current manual system is paper-based, the staff will evidently have varying technical experience as no computer's are required to request an examination, however all NHS workers are trained to have a basic level of computer understanding,

therefore all tasks should be able to be completed without any major problems. There will be five participants, who will take part in the test, as Nielsen (2000) states that elaborate usability tests are a waste of resources and that the best results come from testing no more than five users. Two nurses will complete the test, one radiographer and two doctors. Two nurses have been selected in terms of their technical expertise, Stacey runs the computerised stock system and accounts system, the other nurse Evelyn is not as comfortable with computers, but can do basic tasks. The radiographer, James, has used the manual system previously and is computer literate, however his main role within the department is to schedule and perform examinations, and he is very interested to see for himself how the new system will work. Finally, the two doctors Steve and Andrew have confirmed examination request cards on many occasions and are both of different levels of computer expertise.

7.5.2 Performing the test.

The tests were performed on the same day on a lunch hour, with time allotted to each individual person to conduct the test, this allowed for a one-to-one relationship between examiner and system user, it also provided a more relaxed and un-pressured environment to perform the test. Mayhew (1999) suggests that testing should be “strongly controlled” I feel this was achieved by testing individually. Each user was provided with a short-cut icon to the program and a list of the tasks to perform, the tasks were structured in sequential order of how they would be performed in a real-working scenario, i.e. by first entering patient information, then clinical information, then the exam information. The test was then performed.

7.5.3 Usability Test Results.

This section details a summary of the results of the usability tests, with a full breakdown of the scores in Appendix G along with the questions and tasks posed. Overall, the results were very encouraging and feedback was very positive. The user average rating was 4.4 out of a maximum 5 rating, which indicates the system ease of use, quality of data and user satisfaction. The average rating for each user was slightly lowered by the inclusion of the section on previous system usage, where all five participants had used the paper-based system and rated it, on average, 3.2 for ease of use and 3 for efficiency. The first task required the users to log-in to the system; this was very well received and resulted overall in 5 ratings for every task related to this process. Task two required users to acclimatise to the user interface and to enter patient data, the lowest score in this section was the task relating to entering an appointment time and date, averaging 3.2, with two of the users not knowing what format to enter the date, this was slightly surprising as the inclusion of the tool-tips was expected to solve this question, it was suggested that printing the format above the entry box would be better than a tool-tip for this particular data entry box. Furthermore, the selection of dates posed a problem for users in Task four, with all of the users unaware of the format to enter the date

– from the user feedback it became clear that a tool-tip had not been included for this data entry box and therefore users were unaware if it was the same format as before. Task three required users to enter patient clinical information, these tasks were performed relatively successfully, however there was an average rating of 4 for starting at the root of the clinical help system, which suggested that users were slightly unsure of where the root was, from the feedback obtained, related to the clinical help system, one of the doctors felt there could be more information assigned to different symptoms and clinical information, this response was not a surprise as the system is only a prototype, and requires further expansion, the possibilities for expansion have been outlined in a report to the manager in the system user-guide which can be found in the Appendix at the end of this report. Task four, provided tasks for users related to the entry of examination details, again the tasks performed were relatively successful, however there was a slightly average score of 3.6 for the entry of an examination place, after probing this result further it became apparent that users were slightly unsure of whether a department was required for entry or a hospital name, although there is documentation for users to refer to, this should maybe be clearer for users, a possible solution was to provide a tool tip specifying what is required or by printing above the data entry box what is required, the entry of wrong information would not produce any errors here as alpha-numeric characters were accepted for this field. The final task, five, highlighted three average scores which fell below 4, two of which were to do with the entry of the doctor id and doctor name, however by looking at the results the two doctors gave, their results were both 4's and 5's, and as they would be the only users of the end-system, it was decided that the other users only found the data entry part confusing as they were not doctors and would not know what was required for the doctor id and doctor name. Finally, by comparing the system, to the old card-based system, 5 users found the old system of moderate ease opposed to 3 people finding the new system easier and 2 moderate; furthermore 3 users found the likelihood of making mistakes with the card-system moderate and 2 found it high, opposed to 3 finding it little and 2 moderate with the new electronic system.

7.6 Heuristic Evaluation.

Heuristic evaluation is the most popular of the usability inspection methods, it differs from more conventional experiential usability testing in different ways, one main difference is that evaluators are not drawn from the user community, users are experts in the usability field, and according to Preece et al. (2002), evaluate the user interface driven by a set of usability principles. The objective of using the heuristic evaluation method is to use experts to determine the usability shortcomings of the prototype by envisaging the actions users would take, to uncover any problems that might occur. To help in the evaluation I will use a list of general principles for user interface design, created by Nielsen, called “heuristics” there are ten heuristics, however the general nature of them allows for adaptation depending on the type of interface being evaluated.

7.6.1 Expert Users

As the users are central to this type of heuristic evaluation, it was decided to select a variety of users who could claim to be usability experts, who are very competent with computers and interfaces and also experienced in analysing, testing and using different systems and software. Again, according to Nielsen (2000), an upper limit of 5 experts is recommended, and that the best results come from testing in small numbers and running small tests. It was decided that there should be 3 good evaluators used in the testing of the interface, and according to Nielsen's diagram in the figure below, this would uncover on average 65% of usability problems. Two students studying Computing and Computer Science at Leeds University's school of Computing were chosen to be two of the expert evaluators, who have created different systems and also studied a Human Computer Interaction module at their time at University. I also selected a close family friend who has worked as a computer technician for several years and also works within the Metropolitan Police in charge of their computer department; therefore he is very familiar with this kind of evaluative work.

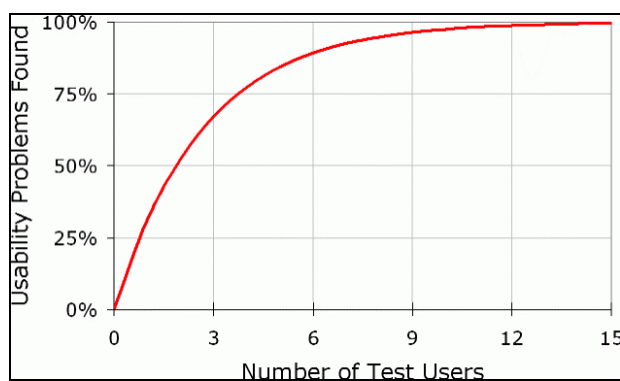


Figure 7.1: Usability Problems to Users – Nielsen (2000)

7.6.2 Performing the test

The list of heuristic principles were developed using Nielsen's ten usability heuristics an improvised version can be found in Appendix H at the end of the report, the expert evaluators were provided with an outline of the user-base to understand the role in which they would be assuming for the test. They were then instructed to follow a list of tasks, relatively quickly to uncover any errors, then re-run paying more attention to detail, every evaluator were given the heuristic set and asked to rate each stage, additional notes were also asked for on the resulting findings.

7.6.3 Heuristic Evaluation Results.

The full breakdown of results can be found in Appendix H at the end of the report, as well as the result averages from all three expert evaluators, this section documents the main points of the evaluation. The heuristic evaluation considered nine different criteria, the first being *visibility*. It was deduced that the interface showed relevant information to the user to be able to complete each task, the interface flowed

well and the response time was short between the user performing a task and the task being executed, one negative point was that highlighting was not used to full advantage, this could have been used to improve visibility.

The *language* of the interface was seen as understandable and simplistic, however the users did find there were one or two technical abbreviations, which may require some further explanation, however the abbreviations were related to the medical examinations and not specific computing terms therefore they would be understandable to the target users.

The *user-control* proved to show one of the lowest scores out of the nine criterions. The two lowest average scores in particular were relating to users being able to exit at any time, within my interface there is no specific exit button, the only exit is by clicking on the cross at the top right of the screen, the experts deemed this an average form of user exit and it also goes against Nielsen's (1994) usability heuristic of not "clearly marking exists". The expert users also felt that the short-cuts for more advanced users was poor, although they did note that the interface was designed in such a way that the tab key on the keyboard could be used to quickly travel from data entry box to the next. *Consistency and Standards* was the next criteria and the experts believed this was satisfied very well as the interface allowed tasks to be performed in the same manner, all the different screens i.e. log-in, main interface and clinical help screen all had the same layout and form, there was also a consistent use of colour throughout.

The system passed the criteria of *recognising errors*, diagnosing them and resolving them, the system would never save or search if there was an error, without producing one for the user, the experts found however that the system was poor at providing immediate error feedback i.e. when entering characters in a number field, you would only find out when trying to save, however when errors were produced they were found to be concise and to the point.

The *prevention of errors* criteria scored an average of 9 out of 15, which I feel is a reasonable score, the main positive point to aid in error prevention was the drop down boxes, this meant that it would eliminate any possible keying errors, however would not prevent a user from selecting the wrong box-selection by accident. The main negative to come from this was there is no confirmation on exiting the system that user really wants to exit, this is a cause for concern as a user could accidentally click the exit button and all information would be lost. The recognition opposed to recall criteria, resulted in two main feedback points, one which was previously known from carrying out the usability test, was that users were not always told the date format which could be confusing, however overall it was deduced that the information on the forms are easy to understand and there was not a lot for a user to actually recall.

The *aesthetic and minimalist design* criteria identified that system could sometimes overload the user with information, as there is a lot to take in on the interface, however the structured nature of this information allowed for it to be managed more easily by the user. It was also deduced that the system was usable to a range of different users, it was suggested that maybe even a disabled user could use the system even though the system was not created for this purpose. Finally, the help information and documentation was seen as excellently created for users in the way that it is task oriented, documents the systems appropriate use and provides a contents page for easy look-up, it was seen that some of the more difficult aspects of the system, namely the clinical help screen, was explained and documented well with screen-shots and not-overly long explanation.

7.7 Card-based manual system vs. electronic requesting system.

One of the evaluation objectives in this chapter was to determine whether the new system is superior to the manual, paper-based process of requesting medical examinations. To determine this, five users took part in a usability test and were requested to use and evaluate the new system and at the end draw comparisons to the manual system, rating the two systems accordingly; the results can be found in the questionnaire in Appendix G at the end of this report. The main outcomes were that the new electronic system was less likely to produce errors, as there are a number of checks in place to prevent erroneous data being added to the database. Furthermore, it was decided that the requesting of examinations across the network was quicker than the manual process of requesting an examination through the use of request cards travelling through internal mail. The main concerns were that errors could be produced when the data entered is not necessarily incorrect, i.e. the date format; this leads to delays whereas with the manual system this can be written and specified quickly. There were also concerns raised about the clinical help system, the actual concept was seen as innovative and useful, however there were questions raised as to whether there would be enough information to support the decision on selecting a medical examination. However, overall users agreed that the quality data the new electronic system offered was a vast improvement to the current manual system.

7.8 Meeting the design and functionality requirements.

This section evaluates whether the system has met the design requirements, the main design requirement was that the system should have a high level of usability for a number of system users; I feel this has been successfully met from the results yielded from the usability test and the heuristic evaluation, on both counts the usability of the system was of a very high standard. The usability test tested five members of the radiology department, who would be actual users of the system, testing involved every aspect of the system testing the users knowledge and whether they knew what they were supposed to do stepping through the system step by step. The average end result of the test was 4.4 out of a maximum 5, giving an overall percentage of 88%, this was a very pleasing result and proved that the system did meet the design requirement of being usable to a number of people. The Heuristic Evaluation that was performed by heuristic experts reinforced this result; the average end result was 74% corresponding to 131 rating out of a maximum 175. The purpose of this test was to uncover any errors and report on whether the system does as it is supposed to, the experts overall were satisfied that this system was appropriate and usable for its intended users, and that if the system could be further improved in the future by considering the negative findings of the evaluation.

Specified in the analysis section of this report were a list of general requirements which the system should aim to meet, obviously not all requirements were expected to be met as the system is a prototype and not fully complete, however the requirements were all fulfilled and the list below details the requirements and how they have been met.

Requirements Met:

1. The system should allow access to examination records and staff has the ability to edit/modify them accordingly.

The interface allows access to examination records by the use of the patient id field and can be accessed directly by clicking on the search button, which will bring-up the information about that patient. Staff then has the ability to edit and update the information if they need to and save back into the database which will perform an update on the tables modified.

2. System needs to be able to support as much clinical information as is needed for the patient.

By using Java's Swing technology, I was able to create a component that allowed the entry of clinical information, which did not restrict the amount of data input.

3. For security purposes, access to the system should be only for registered and authorised personnel.

This objective was met by the creation of a log-in screen which prompts all authorised users to enter their username and password, only users who have been given access to the system will gain entry to the system.

4. System should allow examinations to be requested from any designated patient entry point and received at radiology ready for patient examination.

This is made possible by installing the Examination Request System onto one of the networked computers within the department and sharing it across the network to relevant other departments and/or patient entry points.

5. System should be built in a way which reduces the chance of erroneous or duplicate data being entered and stored into the database.

This objective was met by introducing drop-down boxes with information hard-coded therefore making selection easier without the need to type, thus reducing keying errors. There are also a number of constraints and checks in place both at the database side and interface side, which prevents erroneous information being inserted or updated to the database.

7.9 Evaluating the User-Guide.

One of the minimum requirements of this project was to produce a User-Guide for staff within the department of radiology on correct system usage and as such it was important to deliver this and evaluate it accordingly. I did not provide the guide to staff until after performing the Usability Test and the Heuristic Evaluation, as I wanted to gauge users first impressions of the system and find out its ease of use without the users being guided step-by-step. To Evaluate the user guide, I collected all the negative feedback about the system provided by users who performed the Usability Test, then provided the users with the user-guide, I then asked users who gave negative feedback on aspects of the system to

go through the system again following the user guide and report back on how they felt it helped, if at all. All the users reported back that the user-guide made the whole examination requesting process easier, it also explained the correct formats and explained why error messages were produced and how to rectify the error. One user however reported that it would be impractical to use the guide every time for system use, however also stated that it was ideal for novice users and new NHS staff. In conclusion I can say that the user-guide was successful in fulfilling the stated aim of educating users on the correct system usage.

7.10 Future Improvements.

The system offers a lot of potential for future improvement, many of the improvements fall out of scope of this project, however as detailed in the system-user guide for management, there are lots of enhancements which can be made to different system builds to make the interface even better in terms of improved quality of information. One of the more obvious improvements which could be made is to the clinical support system, a patient can be diagnosed with having a wide range of ailments and symptoms, it is important that the help system captures as much information as possible to be able to suggest a required examination to the highest accuracy possible. Another improvement could be to allow features for management, such as to derive statistics from the database to aid in the creation of reports and annual reviews; this could be incorporated into the current interface by providing a management tab, password protecting it only for management use. Finally, there is a current gap between the examination being confirmed and the actual scheduling for an examination, the staff who carry out the examinations could use a separate part of the system where they can view all the day's examinations in sequential order of time, detailing the durations, technologies and machinery required and any special requirements for that patient; this in itself could be an entirely separate project extension to the Examination Request system.

7.11 Conclusion

Part of this projects aim was to create a system to at least a prototype standard, which simulates the operation of requesting of medical examinations so that users have an enhanced requesting interface, which is quicker than the manual system, to improve the accuracy and save time. I believe that from the evaluation of the system that the aim has been met; the evaluation has used both technical experts and actual end-users of the system, to ensure that the system is usable to its target users, giving two different perspectives and two different approaches to testing and evaluating the interface. Hopefully the outcome of this project will be that the system can be used in the requesting process of examinations and improved upon to enhance the quality.

References

Aniszczyk, C, **Get started now with Eclipse**,

URL: <http://www-128.ibm.com/developerworks/opensource/top-projects/eclipse-starthere.html>

[Accessed: March 2006]

Astrachan, et al. (2000), **Recommendations of the AP Computer Science Ad Hoc Committee**,

URL: <http://www.collegeboard.org/ap/computer-science>

[Accessed: 12 March 2006]

Avison, D E, Fitzgerald, G, (1995), **Information Systems Development: Methodologies, Techniques and Tools**, McGraw-Hill Book Company

Beauchemin, S, (2000), **Software Development Life Cycle Models**,

URL: <http://www.csd.uwo.ca/~beau/CS377/CS377-Development.html>

[Accessed: 9th Feb 2006]

Bennett, McRobb, Farmer, (1999), **Object-Oriented Systems Analysis and Design using UML**, McGraw Hill

Boehm, B, (1998), **A Spiral Model of Software Development and Enhancement** Computer, (Vol. 21, No. 5), pp. 61-72.

Eckstein, R, et al, (2002), **Java Swing**, second edition, O'Reilly

Emprend inc. (2005), **Testing Techniques**,

URL: http://www.projectconnections.com/knowhow/kb_contents/skills/testtech.html [Accessed: 4th April 2006]

Eriksson, HE, Penker, M, (2000), **Business modelling with UML: Business patterns at work**, New York: McGraw Hill

Feigenbaum, B, (2006), **SWT, Swing or AWT: Which is right for you?**,

URL: <http://www-128.ibm.com/developerworks/opensource/library/os-swingswt/?ca=dgr-lnxw01WhichGUI> [Accessed: March 2006]

Fisher, M, (2003), **JDBC API Tutorial and Reference**, Addison Wesley

FOCAS, (2004),

URL: <http://www.osc.state.ny.us/agencies/cas/agencymtg/howtoreadBOM.pdf>

[Accessed: 28th January 2006]

Fowler, M, (2005), **The New Methodology**,
URL: http://www.supremistic.com/development-methodology.html#Waterfall_model [Accessed: 15th Jan 2006]

Gosling, J, (2003), **Swing**, Second Edition, Manning

Jackson, K, (2002), **Choosing Data Gathering Techniques**,
URL: http://www.utas.edu.au/teachingonline/documents/choose_data_techniques.pdf [Accessed: 5th January 2006]

Mayhew, D, J (1999), **Usability Engineering Lifecycle**, 1st Edition, Morgan Kaufmann

Nielsen, J, (1994b), **Usability Inspection Methods**, John Wiley & Sons, New York

Nielsen, J, (2000), **Usability Engineering**, London: Academic Press, Inc.

Nielsen, J, (2003), **Top Ten Mistakes in Web Design**,
URL: <http://www.useit.com/alertbox/9605.html> [Accessed: April 2006]

Norman, K L, (2004), **Levels of Automation and User Participation in Usability Testing**,
URL: http://www.lap.umd.edu/lap/Papers/Tech_Reports/LAP2004TR01/LAP2004TR01.htm
[Accessed: 12th April 2006]

Parker, R, (1997), **Web Design & Desktop Publishing for Dummies**, IDG Books Worldwide

Pincus, M, Miller, R F, (2004), **Running a Meeting That Works**, Third Edition, Barron's Educational Series

Preece, J, Rodgers, Y, and Sharp, H, (2002), **Interaction Design: Beyond Human-Computer Interaction**, John Wiley and Sons Ltd

Rubin, J, (1994), **Design "walk thru"**, **Handbook of Usability Testing**, New York: John Wiley & Sons.

Van Haecke, B, (1997), **JDBC**,
URL: <http://java.sun.com/products/jdbc/overview.html> [Accessed 12th April 2006]

Virdell, M, (2003), **Business processes and workflow in the Web services world e-business Architect**,
URL: <http://www-128.ibm.com/developerworks/webservices/library/ws-work.html>
[Accessed: 18th Feb 2006]

Appendix A

Personal Reflection

This section will reflect upon the entire project, documenting any lessons that can be learned, providing advice to other students who may wish to undertake a similar project so that the same mistakes are not made and the things that went well can be noted. Overall, my project experience has entailed a lot of hard work; however it has been both enjoyable and interesting. All the minimum requirements specified at the beginning of this project were met, and exceeded, by implementing the clinical support interface to aid medical staff in deciphering a type of examination. I am very proud of what I have achieved in such a short space of time. Starting the project was very daunting, however following the waterfall methodology gave me direction and allowed me to manage my time accordingly. It was not however so straight-forward in the beginning as my original project idea of a full business process re-engineering project, within an external company, was dismissed as not being substantial enough to merit a 40-credit project.

However, before the end of the first semester it was decided that I would probably have time to actually be able to create a prototype of the system as well as model the processes; therefore providing more scope to the project. From experiencing this problem I can now speak from hindsight and conclude that if working with an external company, it is important to ensure the work proposed is quantifiable, it is important that the project supervisor and the problem owner within the external company are both happy with the amount of work being carried out, otherwise deadlock can be reached and it is difficult to then move from that position.

Furthermore, when working with an external company one will meet a lot of different people, I wrongly assumed, when I first started the project, that I would only require the help of the actual problem owner, which in my case was the department manager. However, during the analysis phase I needed to speak to all the department staff to understand their role in different processes and to fully understand how the department operates. In the design phase I decided to involve user

participation to look at my screen designs and decide whether they thought it was appropriate. My testing also involved participation from different users to test and verify the system and my evaluation relied heavily on the end-system users and experts to evaluate the system in terms of its improved quality of information and the level of usability. With potentially a lot of projects following a similar approach, it is imperative to structure meetings and work-sessions and account for this time during the project also bearing in mind that scheduling time to conduct these meetings will need to be worked around the company and not the person carrying out the project, the collation and write-up of the results of these meetings and work-sessions can be quite long and a laborious processes.

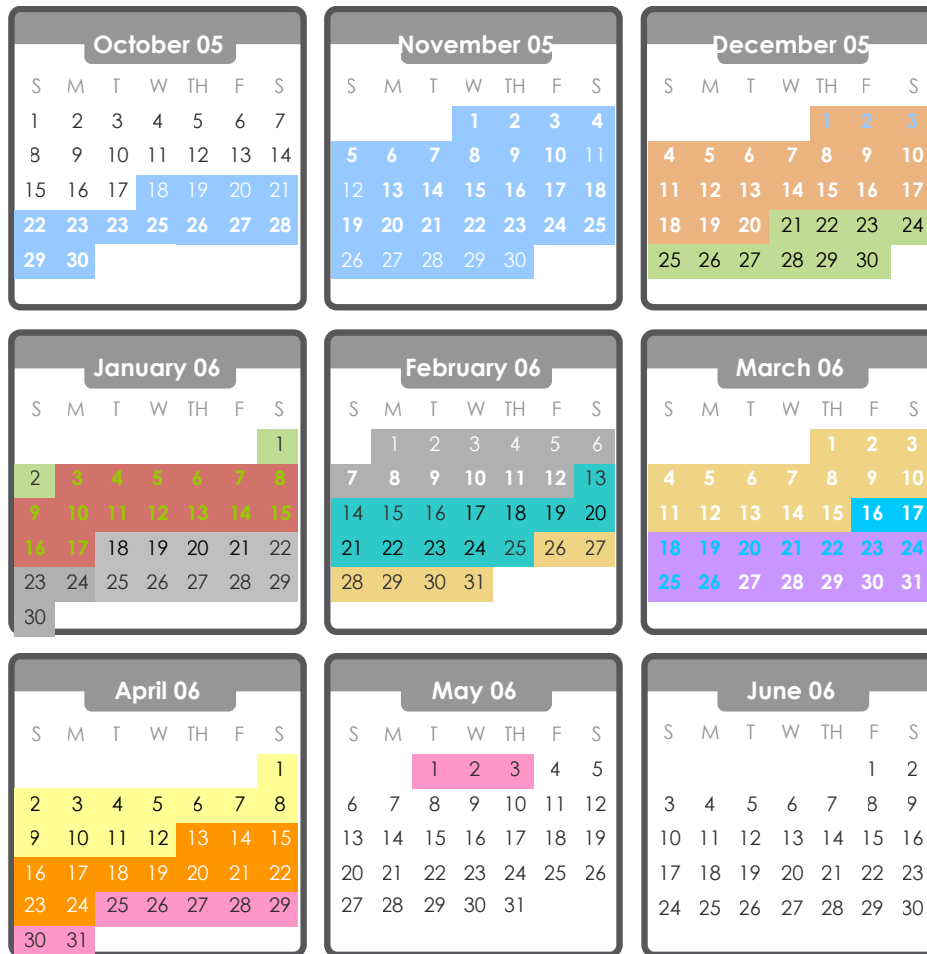
Another factor that became apparent during the analysis was that my proposed system solution incurred some different process changes, ultimately affecting and changing how the department operated; and as my project was conducted within a hospital, the changes were of great significance, because it also affects patients. I was lucky enough to get full backing from the manager who was very enthusiastic and excited about my proposed system and he could immediately see that it would benefit the department; however every single detail and change needed to be recorded. I decided to include a lot of the process changes within the analysis section with more detailed explanation and comparisons drawn within the process report to management, which is included in Appendix D. It is important to consider approaching analysis of this manner when a lot of changes to business process will be incurred as it allows one to understand the full complexity of the problem being investigated and details how one can move from the current state to the new process state. This phase however, was very time consuming, which is why I originally thought of basing my project entirely around modelling the departments different processes and comparing them with proposed changes, however my effective time management and rapid collection of data and information allowed me to carry out the modelling of the processes.

Overall my experience of this project has been very good, with the main positive gained being the feeling of satisfaction from completing a real-life project for a big company such as the N.H.S. My previous project experience had been scenario based, and therefore the same satisfaction cannot be achieved. The fact that this project has allowed me to practice within the field of software engineering has benefited me greatly being an Information Systems student, as I have had time to practice with and learn new technologies. Furthermore, I feel I have developed my communication and time-management skills, from constant contact with staff within the department, which will bode well for me in the future. One of my main personal objectives, was to produce a working solution which had been fully tested and evaluated to a high standard to hand-over to the management, in conclusion I can say I have satisfied this objective to a high-level and I have learned vastly throughout the whole project experience.

Appendix B

Project Schedule

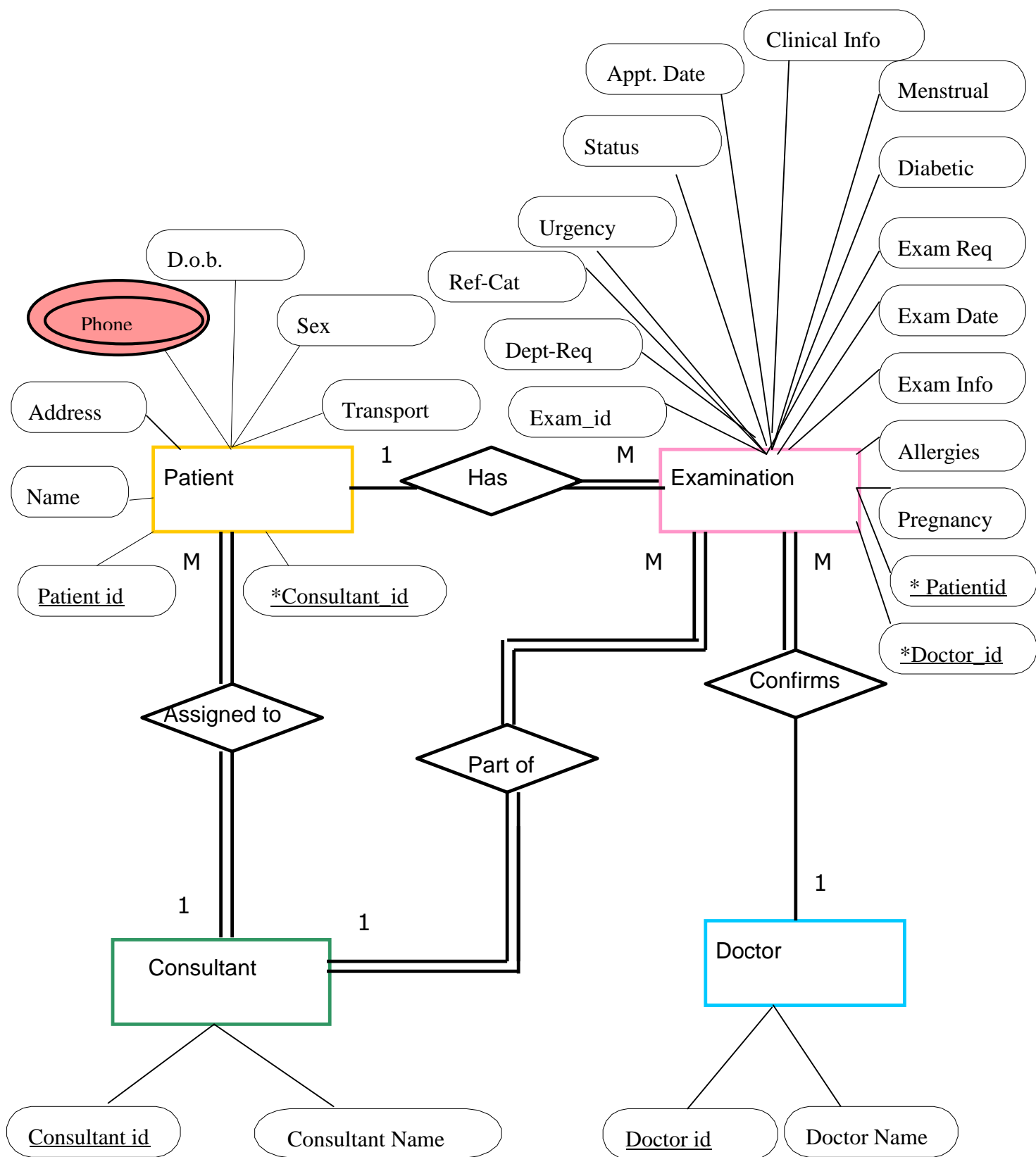
PROJECT EVENT SCHEDULE 05/06



PROJECT PHASE	START DATE	END DATE
Phase 1 – Investigation of the problem	18.10.05	03/12/05
Phase 2 – Write Mid-Project Report	01.12.05	20/12/05
Phase 3 – Revision and taking of exams	21.12.05	17.01.06
Phase 4 –Business-Modeling & Analysis	03.01.06	17/01/06
Phase 5 – Begin work on GUI	18.01.06	12.02.06
Phase 6 – Database set-up and connectivity	13.02.06	25.02.06
Phase 7 – Integrate clinical information help to GUI	26.02.06	15.03.06
Phase 8 – Perform system testing	16.03.06	26.03.06
Phase 9 – Produce relevant system documentation	18.03.06	31.03.06
Phase 10 - Evaluation	01.04.06	13.03.06
Phase 11 – Draft of Final Project Report	14.03.06	24.04.06
Phase 12 – Final Project Report	25.04.06	03.05.06

Appendix C

Complete Entity Relationship Diagram



Legend:

- * : Underline and star indicates foreign key from another table
- _ : Single Underline indicates a primary key
- = : Double association line indicates 'mandatory'

Appendix D

Business Process Report

Process Report

Abstract:

The purpose of this report is to summarise the main process changes with introducing a new electronic system, analysing the results and comparing them with the different business processes associated with requesting an examination using the manual card-based system. The processes will be modelled using established business process modelling techniques using the Unified Modelling Language (UML), presenting the processes in an easy-to-follow, understandable manner. The modelling will capture all the major, daily processes involved with requesting an examination and will allow a conclusion to be drawn upon the benefits and drawbacks.

Card-Based System:

It has been identified through both interview and process analysing that the current card system can be inefficient and wastes resources, by law a nurse can request an examination, however the current system requires a doctor's authorisation, using the current system and bearing in mind the worst case scenario, this can take a number of days to complete. To fully explain the inefficiencies with the card-based system, a business process diagram will be drawn from a high level to model the process of a patient being assessed to the patient being examined, for a successful request, and in the case of an examination not going ahead, the process of a doctor rejecting the request will also be modelled.

Modelling the domain:

Before detailing the processes involved with requesting an examination, it is first important to model the working domain, capturing all the different workers involved in the requesting process. UML can be utilised to model the business workers. By selecting a Business Object Model allows

all the different workers to be captured and modelled, Figure 1.1 below shows this:

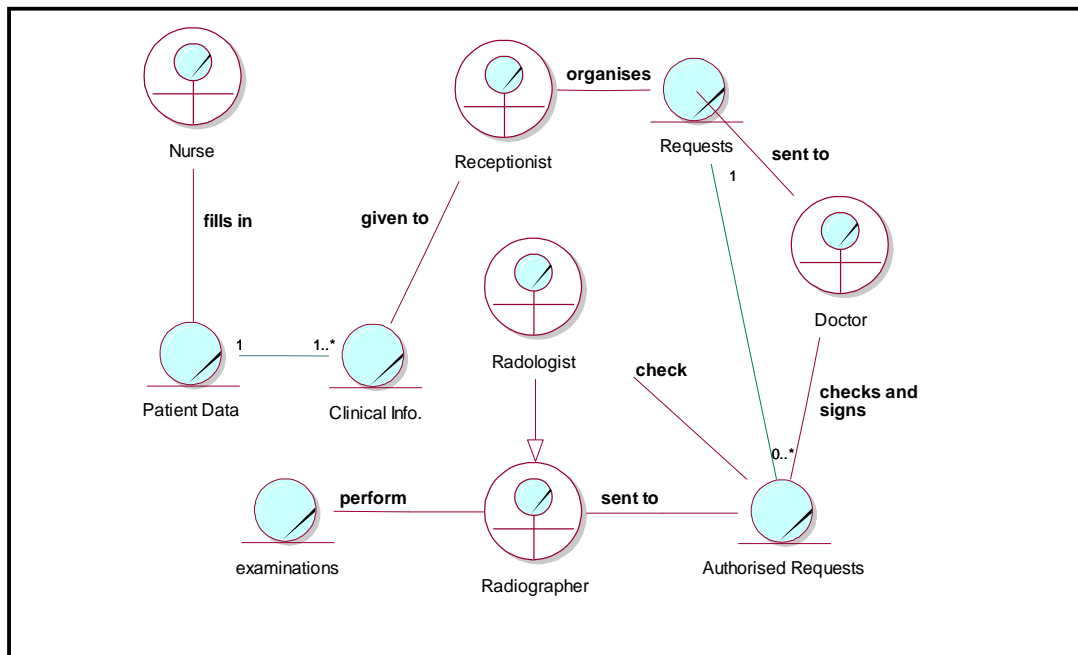


Figure 1.1: Business Object Model of the business workers.

Figure 1.1 details the interactions, which take place between the business workers, and the business objects they are associated with. The diagram shows a nurse filling in patient data and the clinical information relating to a particular patient, the *multiplicity* between the two are highlighted by a green association line, the relationship is 1 to 1 or more, this means one patient can have one or more different parts of clinical information about that patient's condition. The clinical information is then passed onto the Receptionist via the request card, which in turn organises and prioritises the requests, which are then sent to the doctor. The doctor checks and signs the requests if the examinations are to go ahead, these form a number of Authorised Requests, which are then sent to the Radiographer, who will perform the examination, the authorised requests may sometimes be checked by a Radiologist, this business worker has been indicated on the diagram by a generalised arrow from the Radiographer. The multiplicity between Requests and Authorised Requests is highlighted by a green association line, the multiplicity is 1 to zero or more, this is because a number of requests may all not get authorised therefore in the worst case this is zero or more.

The business object model allows for all the business workers to be modelled and the main duties they carry out to link the workers together. However, there are more tasks which were not modelled which each worker carries out on a daily basis relating to requesting an examination. To model the tasks each worker performs, UML supports a business use case diagram which is used to primarily identify the main elements and process that form the current system. The use of actors and processes, termed use cases, help to quickly establish, from a high-level, process ownership.

Business Use Case Diagram:

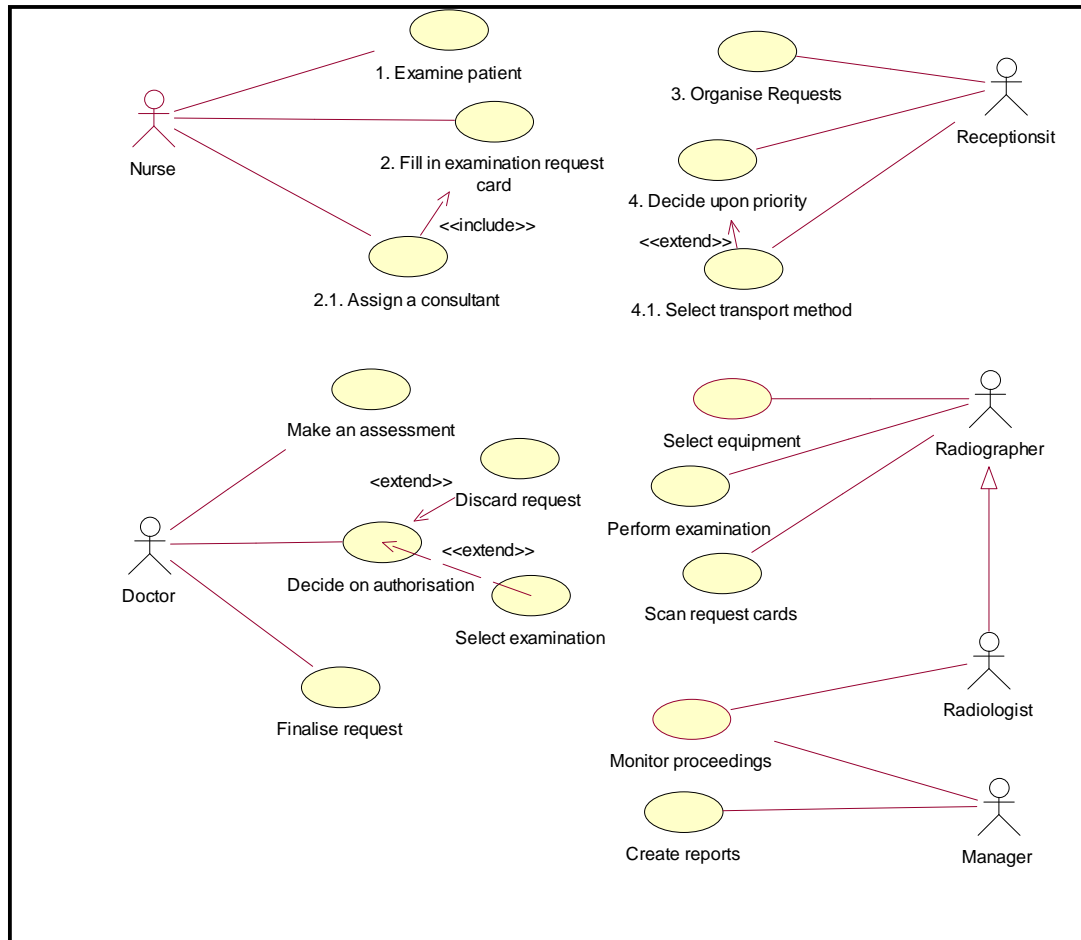


Figure 1.2: Business Use Case Diagram of the department's manual processes.

Figure 1.2 shows all the main duties and processes assigned to the business workers within the department; the processes and duties modelled relate directly to the current card-based system, of course the actors carry out other business activities. The 'include' text used here indicates that the process or duty is included in another use case, for example 'Assign a consultant' is included in 'filling in the examination request', not dissimilarly the term extend is also used, this relates to an expansion of a use case and can be thought of as a parent-child relationship. In the diagram 'Select Examination' and 'Discard Request' are both extensions of the Doctors authorisation decision, of which there are two possible outcomes. Now that the business workers and the duties they perform have been modelled, and understood, using two established modelling diagrams, it is possible to now to introduce a diagram which models the system, showing all the delays involved.

High-Level-Activity-Diagram:

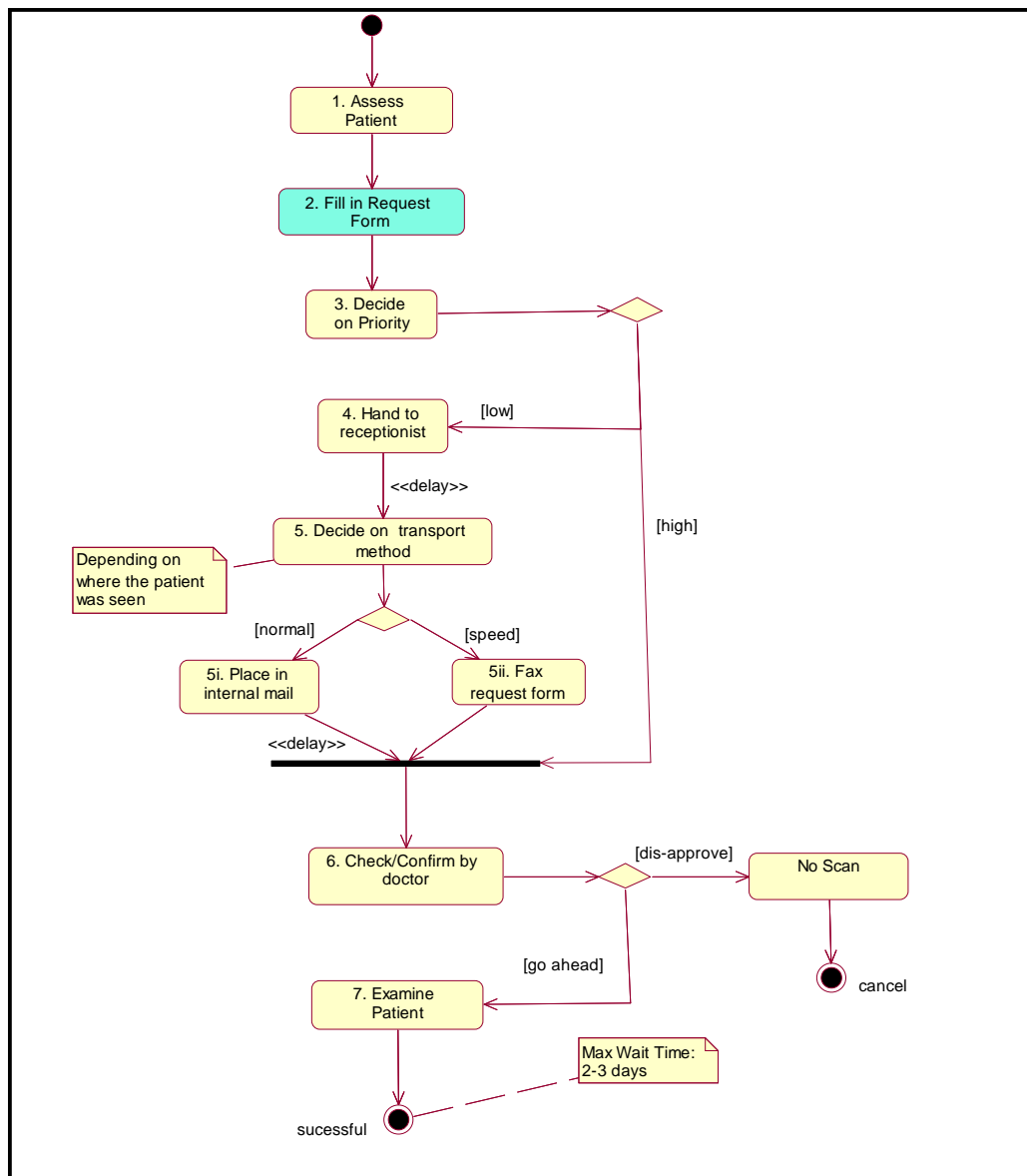


Figure 1.3: High-Level Activity Diagram of the Exam Request Process

Figure 1.3 shows all the possible outcomes from a request being authorised to a request being deemed inappropriate; the diagram has a start point and an end point. The most appropriate format has been chosen for modelling the current system processes.

The figure shows the process flow, from a high level, of the examination request card being filled in following assessment by medical staff; the steps have been numbered for reference purposes. The diagram shows that all requests go through the full sequence of processes 1-5, before a decision is made on whether the examination is to go ahead. This is not necessarily an efficient system, as time is lost getting to process number 6 if an examination is decided not to go ahead. A more

efficient system may not go as far through the sample pathway. Activity number 2 has been highlighted to indicate the main process, which is actually filling out the request form.

Delays:

Also indicated on the diagram are the delays, of which there are two. The first delay is the receptionist choosing the appropriate route and transport method of the request card, this only occurs if the examination request has been deemed a low priority, there can sometimes be a backlog of low-priority request cards hence a delay in the actual sending. The second delay, which is marked, represents the delay experienced by the movement of internal-mail; this can sometimes take a number of days to actually reach a doctor in the worst case. There is no, *real*, delay if the transport method is by fax in the best case, requests can be usually received by the doctor in a number of minutes, providing he or she is in their office at the time of receipt. An electronic system would eliminate both delays and remove the need entirely for the involvement of the receptionist, releasing vital human resources.

Electronic System.

An electronic system would change the way examinations are requested, providing the system has a help facility to aid staff in being able to select an examination, the system would remove the need for every examination request to be confirmed by a doctor, therefore this process would be optional.

Business Use Case Diagram:

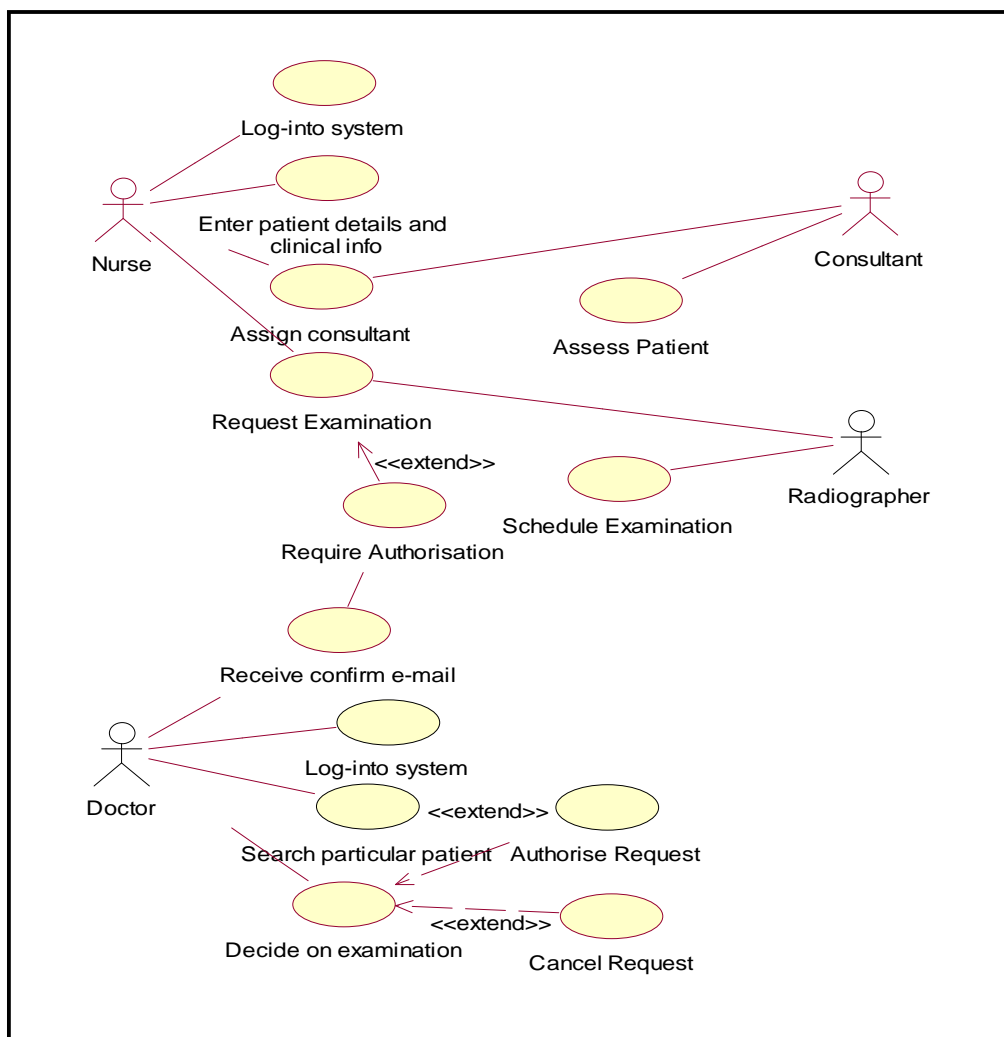


Figure 1.4: Business Use Case Diagram of the staff and processes involved with an electronic system.

Figure 1.4, above, shows a new Business Use Case diagram with staff using the new system. As can be seen, the receptionist is no longer required as the whole process of requesting is carried out electronically. The Manager and Radiologist's participation have been removed to simplify the new processes. The diagram shows a Nurse electronically assigning a consultant to a patient, the consultant then assesses the patient and the Nurse can then proceed with requesting an examination if require, which can be either put through straight away or authorised by sending an authorisation

request to a doctor who receives the request via e-mail. The patient is then searched by the doctor and their details retrieved from the database and displayed, the request is then decided upon and authorised or cancelled. Modelling all the actors and use cases allows for the system to now be modelled in an activity diagram:

Medium Level Activity diagram, with Swim-lanes.

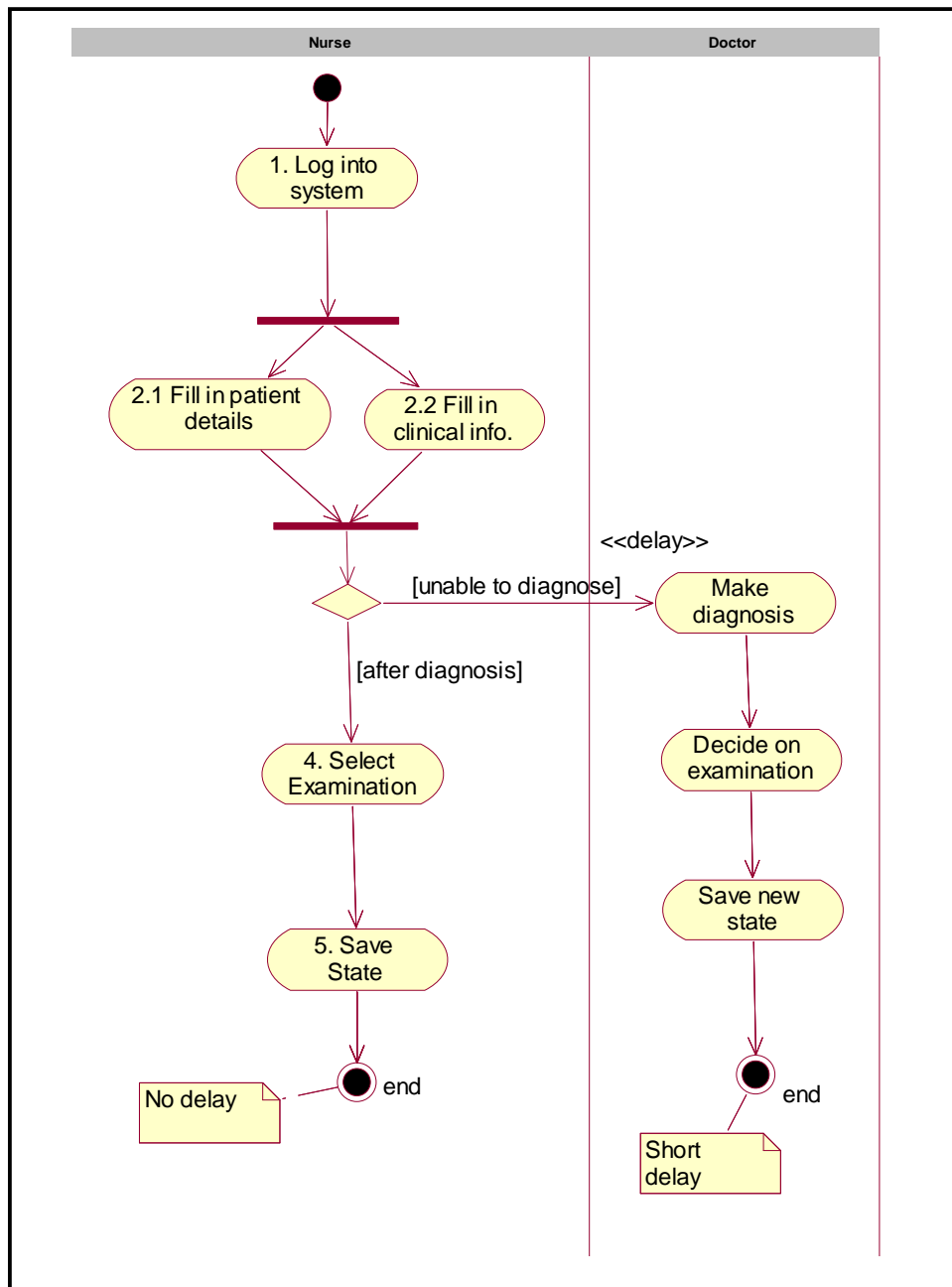


Figure 1.5: Medium Level Activity diagram of the new examination request system

Figure 1.5 clearly shows, via the use of swim lanes, the processes involved with the use of the new system from a medium level. It has a maximum of two users directly involved, and only one user using the system participating in a particular request at any one time.

Delays:

There is no delay if the nurse, or equivalent qualified user, diagnoses the patient and is able to select the examination without needing the doctor's confirmation. There is a short delay with the doctor's intervention; normally examinations will be decided within the same day. The introduction of e-mail mean communication is instant, and the request does not need to travel through internal mail, reducing the chances of mixed-up or lost requests.

Low Level Activity Diagram:

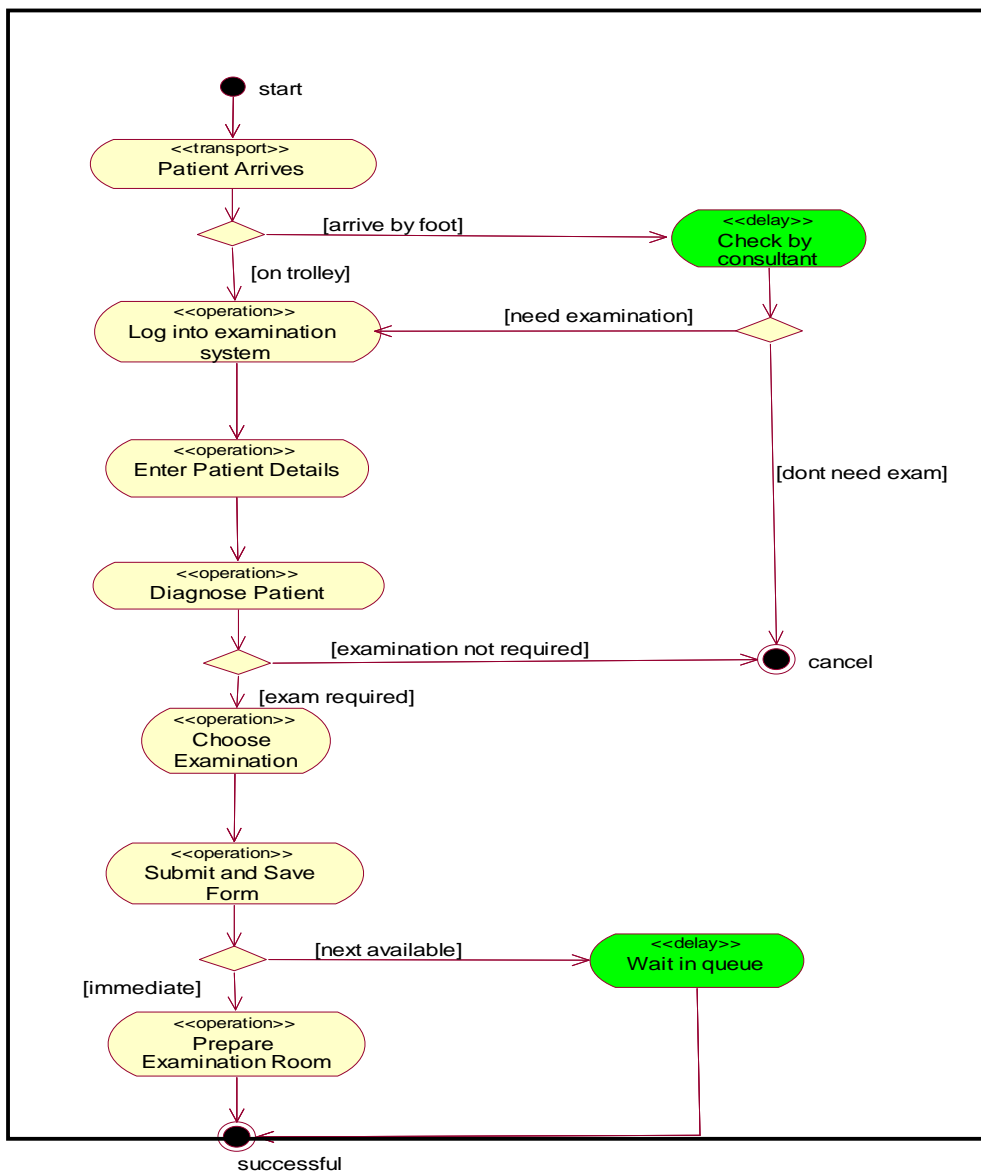


Figure 1.6: Low-Level Activity Diagram requesting an examination with an electronic system.

Figure 1.6, above, shows an activity diagram with an introduction of a new electronic system and the processes involved from a low level, showing the operations closer. The diagram is a more detailed version of activity diagram, using ASME standard, for classifying each activity type. Decisions undertaken are visibly marked using decision diamonds. It is clear that the electronic system removes the associated transport delays that were experienced with the card-based system; however there are still two delays, marked in green, which cannot be helped, even with the introduction of new technology.

Conclusions:

From the modelling of the processes it is clear that from the implementation of the electronic system it will mean a change of departmental business process. Having looked at the different processes involved for the manual card-based system and with the introduction of an electronic system, it can be deduced that the electronic system will save the department both time and resources in a number of ways. Firstly, resources are freed by removing the need for the receptionist completely, allowing the receptionist to do other departmental tasks and operations. Secondly, the doctor is also not required in every instance to confirm an examination, providing that there is enough information available for a nurse or qualified member of medical staff to be able to decide upon the correct examination required. The use of the activity diagrams allowed for time delays to be estimated and placed on particular processes within the diagram, for the manual system there is a maximum waiting delay for an examination being confirmed of two to three days, whereas in the best case with the electronic system the request is instant and in the worst case it is dependant upon the doctor checking his/her e-mail inbox, but doctor's check periodically throughout the day, therefore the maximum wait would be a few hours for a "low priority" request.

Appendix E

Testing

GUI Functionality Testing:

This section includes a table of the testing procedures performed on every aspect of the system GUI, including all the input boxes and action buttons, to ensure the expected result was achieved.

Tested Area	Input	Expected Result	Actual	DB Check	Problem Resolved
<i>Exam System:</i> Testing exam data goes into database Exam Table	Valid Exam Details	Inserts exam details into the Exam Table	<i>Confirmation Message:</i> “Data has been added into the database”	Data has been added correctly	N-A
<i>Exam System:</i> Testing the search button returns an existing patient stored in the DB.	Valid Patient ID number	Patient details and related examination details are returned to the system.	Patient Returned and rows displayed.	Patient returned matches the patient stored	N-A

Tested Area	Input	Expected Result	Actual	DB Check	Problem Resolved
<i>Login:</i> Testing correct username and password results in successful login	sean sean	Login to system	Login to system	N/A	N/A
<i>Login:</i> Testing incorrect username and password results in log-in fail	test1 test2	Log-in failure	Log-in failure	N/A	N/A.
<i>Exam System:</i> Testing the search button doesn't return a patient not in the database	Invalid Patient ID number	Error Exception Thrown – patient is not in the database	Error Exception – Patient ID does not exist	The patient ID is not in the database	N-A
<i>Exam System:</i> Testing the phone number field with an incorrect format	A letter	When attempting to add to the database will return with an error	Violation Error – cannot add to the patient table	Was not added to the patient table	N-A
<i>Exam System:</i> Testing the DOB field with an incorrect format	999999	When attempting to add to the database will return with an error	Violation Error D.O.B cannot add to the patient table	Was not added to the patient table	N-A
<i>Exam System:</i> Testing the DOB field with a correct format	01-11-92	When attempting to add to the database will return will be successful	<i>Confirmation Message:</i> “Data has been added into the database”	Data has been added correctly	N-A

Tested Area	Input	Expected Result	Actual	DB Check	Problem Resolved
<i>Exam System:</i> Checking SEX field with wrong letter	G	When attempting to add to the database will return with an error	Violation Error SEX cannot add to the patient table	Was not added to the patient table	N/A
<i>Exam System:</i> Checking SEX field with correct letter:	M	When attempting to add to the database will return will be successful	<i>Confirmation Message:</i> "Data has been added into the database"	Data has been added correctly	N/A.
<i>Exam System:</i> Checking Transport field with correct letter	Y	When attempting to add to the database will return will be successful	<i>Confirmation Message:</i> "Data has been added into the database"	Data has been added correctly	N/A
<i>Exam System:</i> Checking Transport field with incorrect letter	?	When attempting to add to the database will return with an error	Violation Error Transport cannot add to database	Was not added in the DB	N-A
<i>Exam System:</i> Checking doctor_id with correct number stored in DB	1	When entered, the doctor's name will be displayed	Doctor's name was displayed correctly	It was the correct doctor	N-A
<i>Exam System:</i> Checking doctor_id with incorrect no.	5	When entered nothing will be displayed in doctor name.	No doctor name was displayed	No doctor stored under that number	N-A
<i>Exam System:</i> Doctor confirmation correctly sends email check	N/A	When checked an email should be sent to stored e-mail address	E-Mail was sent upon click	N/A	N/A

Tested Area	Input	Expected Result	Actual	DB Check	Problem Resolved
<i>Exam System:</i> Correctly confirming exam	N/A	When check confirm exam button it confirms an examination	Exam confirmed	Boolean = True	N/A
<i>Exam System:</i> Correctly Entering patient name	David Johnson	Will get added to the database without any problem	Added	Name is there	N/A
<i>Exam System:</i> Testing the Exam Date with incorrect format	999999	When attempting to add to the database will return with an error	Violation Error Exam Date cannot add to the Exam table	Was not added to the exam table	N-A
<i>Exam System:</i> Testing the Exam Date with correct format	01-10-06	When attempting to add to the database will return will be successful	<i>Confirmation Message:</i> "Data has been added into the database"	Data has been added correctly	N-A
<i>Exam System:</i> <i>Testing AM/PM field with correct format</i>	PM	When attempting to add to the database will return will be successful	<i>Confirmation Message:</i> "Data has been added into the database"	Data has been added correctly	N-A
<i>Exam System:</i> <i>Testing AM/PM field with incorrect format</i>	PS	When attempting to add to the database will return with an error	Violation Error AM/PM cannot add to the Exam table	Was not added	N-A

J-Unit Testing:

This section includes some print-screens of some sample Java code when performing a JUnit test on the Examination Request main interface.

```
/*
 * Testing the Exam GUI - testName
 */
public final void testName() {
    Name = "Dave Parish";
    assertEquals(("Paul Parry"), Name);
}
```

Figure 1: Testing the Name entry field for an expected error [Image taken inside Eclipse].

Figure 1 above is a screen-shot of Java code from the JUnit examTest class. This is the typical style of Unit testing, writing test code to intentionally get an error. Above, the method is taking the “Name” String field from the exam.java main class, and assigns the name as “Dave Parish. The following line of code makes an assertion that “Paul Parry” is equal to that of Name, which is obviously not true, therefore an error should ensue. The figure below shows a screen shot of a test failure when testing the getName method:

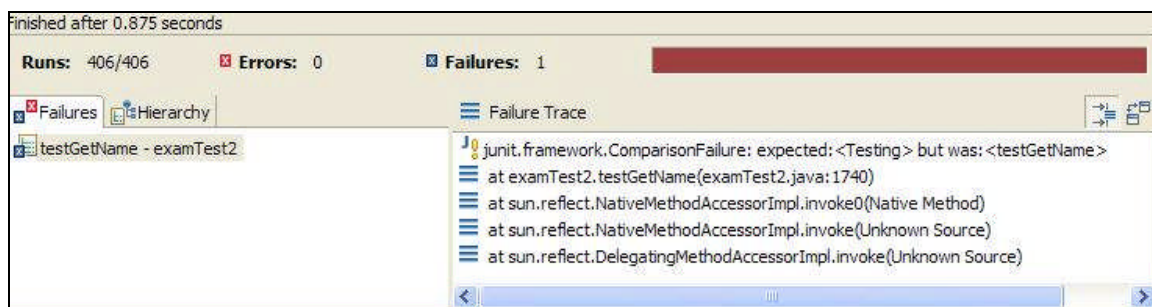


Figure 2: Test Failure when testing the “Get Name” method. [Image taken inside Eclipse]

This example is one of the simpler methods taken from the Test class to show how this form of regressive testing was performed throughout the implementation phase of the project. The purpose of the testing is to make assertions to get the actual and expected coding results, similar to GUI, non-code, testing above, however the errors are not a bad thing and help in assessing whether the code does as the user requires and if not provides suggestions for code improvement.

Appendix F

User Manual

**Examination Request System for the
Clinical Radiology dept. at Leeds
General Infirmary**
Sean Webster
User Manual
2006

Contents

1	Introduction	5
	1.1 Overview	5
	1.2 Logging in	5
	1.3 Logging out	6
2	Examination Request Interface	7
	2.1 About the interface	7
	2.1.1 Tool Tips	8
	2.1.2 Drop-down boxes	8
	2.1.3 Check-boxes	8
	2.1.4 Text Fields	9
	2.1.5 Buttons	9
3	Adding Patient Information	10
	3.1 Importance of correct patient information	10
	3.1.1 Patient Contact Details	11
	3.1.2 Department Required	11
	3.1.3 Reference Category	11
	3.1.4 Urgency	11
	3.1.5 Status	12
	3.1.6 Appt. Date	12
	3.1.7 Transport	12
	3.2 Consultant Assignment	13
4	Clinical Information	14
	4.1 Entering patient clinical information	14
	4.2 Using the clinical help feature	14
	4.3 Diabetic Information	16
	4.3.1 Pregnancy Information	16
5	Previous Examination Details	17
	5.1 Previous Examination	17
	5.1.1 Date of Examination	17
	5.1.2 Place of Examination	18
	5.1.3 Allergies to contrast medium	18
	5.2 Examination Requesting	18
	5.3 Doctor Confirmation Required.	19
6	Doctor Confirmation	20
	6.1 Confirmation by a doctor	20
	6.2 Searching the patient number	20
	6.3 Entering unique doctor id	21
	6.4 Changing Suggested Exam	21
	6.5 Entering any concluding notes	22
	6.6 Confirming the exam and saving	22

7	General: Saving data and searching and querying	23
	7.1 Saving	23
	7.2 Searching	23
8	Management Section	24
	8.1 How the system could be extended	24
	8.2 Changing the code	24
	8.3 Drivers	25
	8.4 Recommendation.	26

Chapter 1

Introduction

1.1 Overview.

This manual details the appropriate usage for all aspects of the Medical Examination Request System. The guide is divided into nine chapters. All users should read this first chapter before proceeding to use the system; it details how to log-in and log out of the Examination System. The following seven chapters are recommended for all potential users of the system, with the final chapter for the attention of the manager only. Chapter two describes and introduces the main Examination Request interface and its related components. Chapter three shows how to add patient information into the system and chapter four details how to make use of clinical information for a particular patient. Chapter five describes the importance of a patients previous examination details and how this should be correctly entered. Chapter six details the involvement of doctors to confirm an examination and explains why this is necessary and how they go about doing this by using the examination system. Chapter seven details how to save and retrieve information from the database with the final chapter for management only.

1.2 Logging in.

The login window is shown in figure 1 below:



To login:

1. Enter your assigned username into the 'Username' field.
2. Enter your assigned password into the 'Password' field.
3. Left click on 'Login'.

If your user-name and password do not match those stored in the database, or it is that you are not an authorised to use the system, access will not be granted.

1.3 Logging out.

To log out of the system is simple, there is no log-out system currently in place, all that is required is for the user to click the red cross button located in the top right of every screen, this will then log a user out at any time, it is important to stress that all changes should be saved before logging-out, because all unsaved data is lost when a user log's out.

Chapter 2

Examination Request Interface

2.1 About the Interface.

The interface has been programmed in a programming language called Java, which will run on all the platforms within the department and throughout other departments in the hospital if required. The features which need more explicit explaining are the drop-down boxes, test fields, check-boxes and buttons – their inclusion in the GUI are explained below, the main system interface can be viewed below in Figure 2.

Examination Request System

Patient Info:
Name: ID:
Address:
Tel No:
Consultant:
DOB: SEX: Transport:

Clinical Information:

Diabetic: Yes ☐ Possible Pregnancy: Yes ☐ Proceed if period o/due: Y ☐

Examination Details:
Previous Exam: Yes ☐
Date:
Place:
Allergy to Contrast Medium: Yes ☐
Exam Requested: CT Scan ☐
Doctor Conf. Required: ☐

Doctor Confirmation:
Doctor Id:
Doctor Name:
Exam Date:
☐ Examination Confirmed
Doctors Notes....

Figure 2: Examination Request GUI

2.1.1 Tool Tips.

Within the Interface are pre-programmed tool tips for users to make explanation of what is required for entry a little simpler. By hovering over a field or component will bring up a tip for the user, indicating usually by hashes the style of entry and the length, this is helpful for first time users of the system or infrequent users who cannot remember what the format for entry is, an example of a tool tip can be seen in Figure 3.1 below:

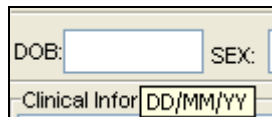


Figure 3.1: Tool Tip for D.O.B.

2.1.2 Drop-Down Boxes.

The interface introduces another new component, in the form of drop-down boxes, it is important to understand that the data within these boxes are unalterable, the benefit of this is to reduce typing errors and also to save time, to make a selection it is required that the user left click anywhere on the relevant drop-down box, then scroll down and select the appropriate attribute. Figure 3.2 below shows a user making a selection on the 'urgency' of an appointment.

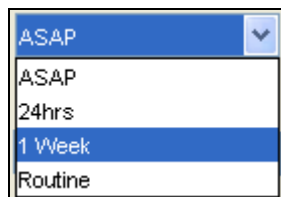


Figure 3.2: Drop-down box for Urgency field.

2.1.3 Check-Boxes.

Within the interface are components called check-boxes, these are used where a confirmation is needed, by ticking the check box this is activating it for a 'Yes' or 'True' leaving the check box blank is leaving it in a 'No' or 'False' state. They are used within the interface twice, to confirm an exam, the figure below identifies the check box and sets it to a 'true' state:



Figure 3.3: Check-Box requiring a doctor's confirmation.

2.1.4 Text Fields.

There are two types of text field within the interface, one is restrictive of the amount of data the field can hold the other is unrestrictive and can be extended by the use of scroll bars. The normal text fields are to take names, addresses, dates and single letters, whereas the extended text fields can hold large amounts of clinical information or doctors notes. The two Fields can be seen in Figures 3.4 and 3.5 below:

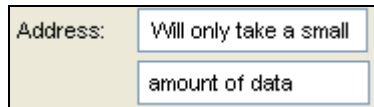
A screenshot of a form with a label 'Address:' and a text input field. The text inside the field is 'Will only take a small amount of data'.

Figure 3.4: Text field for the address.

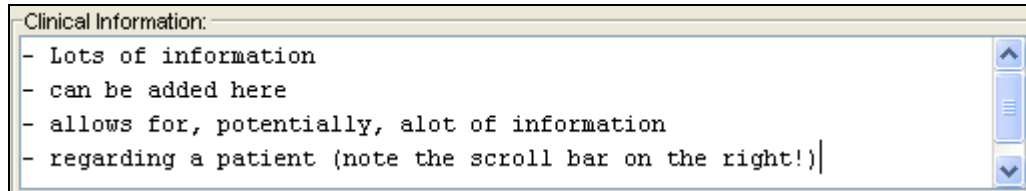
A screenshot of a form with a label 'Clinical Information:' and a text area. The text inside the area is '- Lots of information', '- can be added here', '- allows for, potentially, alot of information', and '- regarding a patient (note the scroll bar on the right!)'. There is a vertical scrollbar on the right side of the text area.

Figure 3.5: Text Field for the patient clinical information.

2.1.5 Buttons.

There are buttons within the form that when pressed perform a certain action, there are three on the interface, the 'clinical help' button when pressed will bring up a new window to help in linking the clinical information about a patient to an examination suggestion, the 'search' button is used when searching for a patient is required, and the 'save' buttons saves the current state of the interface to the database, there is more on searching and saving later in this guide. Figure 4, below, shows a screen-shot of the search and save buttons:

A screenshot of two buttons side-by-side. The left button is labeled 'Search' and the right button is labeled 'Save'.

Figure 4: Search and Save buttons.

Chapter 3

Adding Patient Information

3.1 Importance of correct patient information.

It is imperative that the patient information entered is correct, as incorrect information can have an effect on the urgency of the patient's case and can lead to unwanted delays, within the National Health Service time is precious and accuracy is always of paramount importance. The interface allows for information about a patient to be displayed in a readable and structured manner, it also helps prevent the rate of errors by introducing fixed drop-down selections. However, typing is still required therefore it is important that users make sure there are no errors, this section of the user manual will explain every part of entering a patient information to ensure users are educated thus hopefully preventing any serious errors from occurring.

3.1.1 Patient Contact Information.

Entering the patient's contact information is the first action which needs to be performed when logged into the system. The patient information is made up of a series of text boxes, the patients full name goes into the "Name" field, and their related patient id into the "ID" field, if the patient ID is not known then a unique numerical id for needs to be created and stored for that patient for future retrieval. The address and telephone number for that patient follow, with the address line one in the first text field and line two in the second field. The telephone number should be in the format of '0000-0000000' to be accepted. Figure 5.1 shows the patient info screen below:

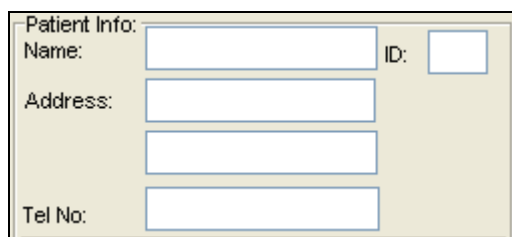
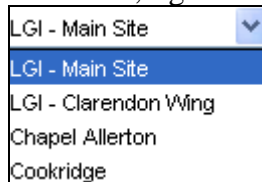
A screenshot of a web form titled "Patient Info:". The form contains several input fields: a "Name:" field, an "ID:" field, an "Address:" field with two stacked text boxes, and a "Tel No:" field with one text box. The form has a light beige background and a thin border.

Figure 5.1: Patient Information Box.

3.1.2 Department Required.

Once the personal details of the patient have been entered, it is then important to work through the list of pre-required information for that patient. The first is to decide on the department required for that patient for consultation, two of which are within the hospital and two are departments in other local hospitals. There are four possible departmental selections, figure 5.2 below shows this, with the current selection set to 'LGI Main Site'.

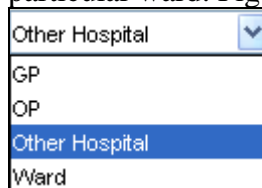


A screenshot of a dropdown menu titled 'Department Required'. The menu is open, showing five options: 'LGI - Main Site' (highlighted in blue), 'LGI - Main Site', 'LGI - Clarendon Wing', 'Chapel Allerton', and 'Cookridge'. The first option is also the current selection shown in the dropdown arrow.

Figure 5.2: Department Required

3.1.3 Reference Category.

The reference category refers to who the patient should be referred to or for what they should be referred to. There are four choices, the GP should be selected if the patient is to be referred to the General Practitioner, OP should be selected if the patient be referred for an operation, other hospital should be selected if the department or the hospital do not have the required personnel available and ward should be selected to refer a patient to a particular ward. Figure 5.3 below shows the contents of this box.

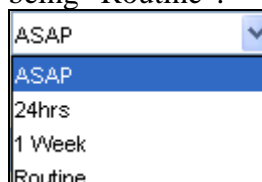


A screenshot of a dropdown menu titled 'Reference Category'. The menu is open, showing four options: 'Other Hospital' (highlighted in blue), 'GP', 'OP', and 'Ward'. The first option is also the current selection shown in the dropdown arrow.

Figure 5.3: Reference Category Box.

3.1.4 Urgency.

The urgency of the patients case should be assessed by the user and a decision made as to the urgency of their status, there are four choices the quickest being "ASAP" to the slowest being "Routine".



A screenshot of a dropdown menu titled 'Urgency'. The menu is open, showing four options: 'ASAP' (highlighted in blue), '24hrs', '1 Week', and 'Routine'. The first option is also the current selection shown in the dropdown arrow.

Figure 5.4: Urgency Box.

3.1.5 Status.

The status refers to how the patient has arrived; it may be that they are in a bed, in a chair or on a drip, or even walked in. It is important to provide this information as it can sometimes decide on the priority of consultation if the patient. Figure 5.5 below shows the status box when expanded:

A dropdown menu with a blue header bar. The list of items includes 'Bed' (highlighted), 'Bed', 'Chair', 'Cot', 'Drip', 'o2', 'On Ward', 'OT', and 'Trolley'. Arrows are visible at the top and bottom of the list.

Figure 5.5: Status Box.

3.1.6 Appointment Date.

The appointment date specifies the date of the appointment, from the current day. If routine is selected it could be that the appointment date is 2 weeks, the format for entering an appointment date is DD/MM/YY and then the AM/PM box specifies that the appointment is due to take place in the morning between the hours of 9am-11-59am or 12pm – 4pm.

A form element with a light beige background. It contains the label 'Appt. Date:' followed by a white text input box, and 'AM/PM:' followed by another white text input box.

Figure 5.6: Appointment Date.

3.1.7 Transport.

The transport box exists to decipher whether transport is needed for a patient, this only applies if the patient is referred to another hospital, in this case the transport would box should read “Y” otherwise the box should read “N”. The box does accept Null values, however it is recommended to always specify either Yes or No. Figure 5.7 shows the transport field filled in with a “Y” value.

A form element with a light beige background. It contains the label 'Transport:' followed by a white text input box containing the character 'Y'.

Figure 5.7: Transport Box.

3.2 Consultant.

The consultant field is for the user to provide the full name of the consultant which a patient is to be referred to. Figure 6 below shows the consultant field filled in with a name of a consultant.

A form element with a light beige background. It contains the label 'Consultant:' followed by a white text input box containing the text 'Mr A. Palma'.

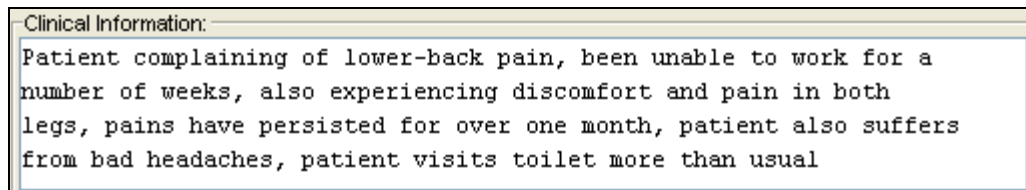
Figure 6: Consultant Box.

Chapter 4

Clinical Information

4.1 Entering Patient Clinical Information.

Entering correct patient clinical information is imperative to being able to decide upon whether the patient requires a medical examination. The key is to enter as much key information regarding the patient's current condition as possible, keeping to the main ailments and problems and also including the time scale with which the patient has been suffering. Figure 7 shows an example of a patient who has arrived complaining of lower back and leg pain and the amount of time they have had this for.



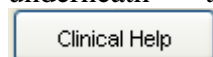
Clinical Information:

Patient complaining of lower-back pain, been unable to work for a number of weeks, also experiencing discomfort and pain in both legs, pains have persisted for over one month, patient also suffers from bad headaches, patient visits toilet more than usual

Figure 7: Patient Clinical Information Form.

4.2 Using the Clinical Help Feature.

The clinical help feature can be initialised by clicking on the “Clinical Help” button underneath the Clinical Information, the button looks like this:



Once clicked it will then bring up a new screen called the “Clinical Information Help System” the purpose of this system is to use what has been collected about the patient and link this together by following a series of steps, if there is not enough clinical information to make a decision, then more can be requested from the patient's, or a doctor can make a decision on whether an examination is required. The Clinical Help screen can be viewed below in figure 8 along with a walkthrough of what is displayed.

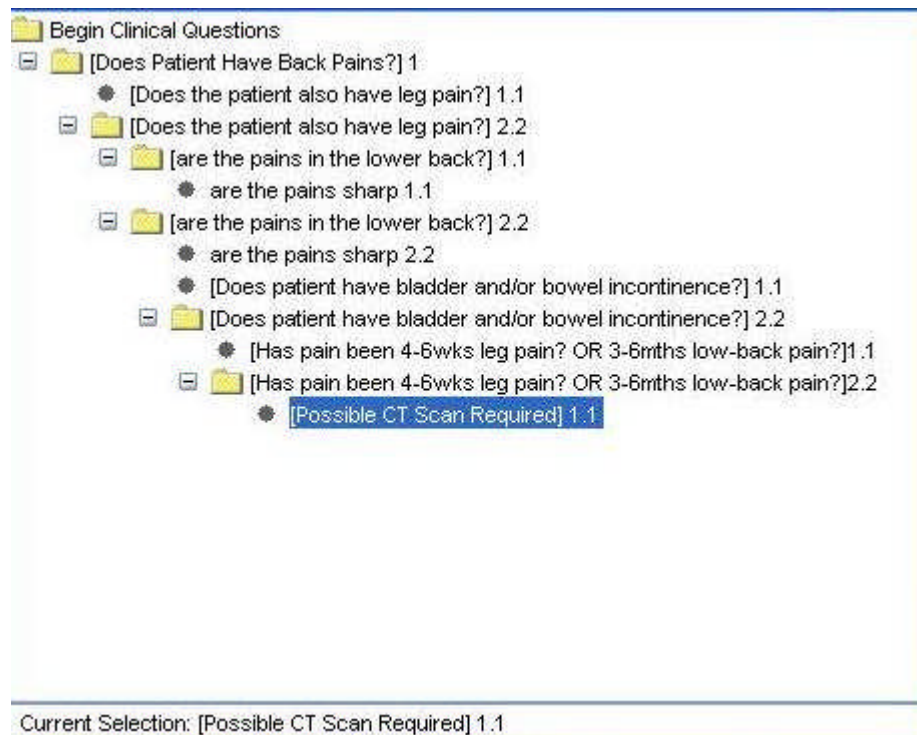


Figure 8: Clinical Information Help Screen.

The system starts by deciding upon a clinical question to start with, which will contain a series of linked symptoms and will end with a suggestion for a type of examination. From the clinical information on the main interface, it is apparent that the patient is suffering mainly from back pains, by clicking on “does the patient have back pains” the system knows that “leg pain” is commonly associated with having back pain, therefore another question is asked, once clicked on it brings up two folders 1.1 and 1.2 both ask different questions, however the one most relevant to the patient is 2.2 which asks whether the pains are sharp and/or does the patient have bladder or bowel incontinence, from the clinical information it was highlighted that the patient was visiting the toilet more often than usual, therefore the user proceeds with more questions related to this folder. It then asks on the time scale of the pains for both the leg and/or the back, we know from the clinical information that the patient’s leg pain has persisted for over a month, therefore the user proceeds with the questions, the series of questions have now finished in this instance and the final node suggests that a possible CT Scan is required. Depending on the severity of the patient’s condition the exam can be scheduled to go ahead or a second opinion can be sought from a doctor. However, this system allows qualified users to make more sense of clinical information and allows them to make a decision on a patient without any delay.

4.3 Diabetic Information.

The diabetic information box is simply to tell the nurse or doctor whether the patient suffers from diabetes, as this may or may not have an effect on the type of examination they require. From the drop down box the user can either select “yes” or “no”, the default is set to “no”.

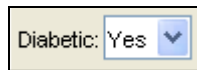
A screenshot of a web form element. It consists of a label 'Diabetic:' followed by a dropdown menu. The dropdown menu is open, showing the selected option 'Yes' and a small downward-pointing arrow icon.

Figure 9.1: Diabetic Information.

4.3.1 Pregnancy Information.

The pregnancy information is to tell the nurse or doctor whether the symptoms they are experiencing may be a cause of a possible pregnancy; therefore an examination may not be appropriate if a patient is in an early stage of pregnancy. Furthermore, the second drop-down box indicates whether or not the exam should proceed if a women's menstrual period is overdue., the default for this field has been set to N/A however it is possible to select either "Yes" or "No" N/A should apply as default if the possibility of pregnancy is set to "no".

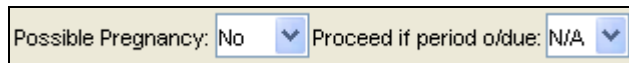
A screenshot of a web form containing two dropdown menus. The first dropdown is labeled 'Possible Pregnancy:' and has 'No' selected. The second dropdown is labeled 'Proceed if period o/due:' and has 'N/A' selected. Both dropdowns have a small downward-pointing arrow icon.

Figure 9.2: Pregnancy Information.

Chapter 5

Previous Examination Details

5.1 Previous Examination.

The previous examination information is important as it will indicate where and when a previous examination took place. The previous exam box is a field which simply accepts “Yes” or “No”, the default for the box is set to “No”. Figure 10.1 shows a screen-shot of this aspect of the system.

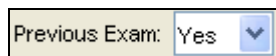
A screenshot of a web form element. It consists of a label 'Previous Exam:' followed by a dropdown menu. The dropdown menu is currently set to 'Yes' and has a small downward arrow icon on its right side.

Figure 10.1: Previous Examination Box.

5.1.1 Date of Examination.

The date of examination field specifies the date the patient previously had an examination, obviously this field should be left blank if no previous examination took place, and the format the date should be in is the standard ‘00 /00/1900’ format.


A screenshot of a web form element. It consists of a label 'Date:' followed by a text input field. The input field is currently empty.

Figure 10.2: Date of Examination.

5.1.2 Place of Examination.

The place of examination field specifies the place the previous examination took place, again if no examination took place, this should be left blank, the database accepts a null value for this field.

A screenshot of a web form element. It consists of a label 'Place:' followed by a text input field. The input field is currently empty.

Figure 10.3: Place of Examination.

5.1.3 Allergies to contrast medium.

A contrast medium is a substance that enhances information contained in an images produced by medical diagnostic equipment, for examinations such as digital radiology, nuclear medicine and ultrasound. Some patients can be allergic to the substances used, therefore it is important to report whether or not a patient has history of any allergies to these media. The default for this field is set to “No”, however it is important to change this if the patient is allergic, the figure below shows this field.

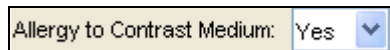
A screenshot of a web form element. It consists of a text label "Allergy to Contrast Medium:" followed by a dropdown menu. The dropdown menu is currently set to "Yes" and has a small downward arrow icon on its right side.

Figure 10.4: Allergy to contrast medium.

5.2 Examination Requesting.

This is the part of the interface where an examination is requested, there are five different choices to choose from, there is no default set, if an examination cannot be decided upon then it should be left.

A screenshot of a web form element. It is a dropdown menu with a blue header bar containing the text "CT Scan" and a small downward arrow icon. The menu is open, showing a list of options: "CT Scan", "MRI Scan", "Nuclear Medicine", "Radiogrpahy", and "Ultrasound".

Figure 11: Examination Requesting Box.

5.3 Doctor Confirmation Required.

If the examination cannot be decided upon then the user should click the doctor confirmation check-box, shown in Figure 12 below, this will activate an e-mail to a doctor which will send the patient id concerned, the doctor can then confirm the examination.

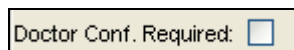
A screenshot of a web form element. It consists of a text label "Doctor Conf. Required:" followed by an unchecked checkbox.

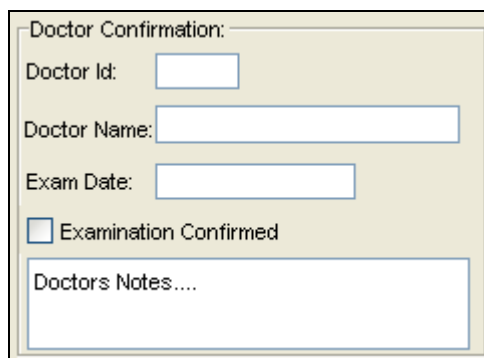
Figure 12: Doctor Confirmation Required.

Chapter 6

Doctor Confirmation

6.1 Confirmation by a doctor.

Some examinations may require a doctor to confirm an examination if an examination selection cannot be decided upon given the clinical information provided by the patient. The doctor confirmation screen is located at the bottom right of the interface and can be seen in Figure 13 below:



Doctor Confirmation:

Doctor Id:

Doctor Name:

Exam Date:

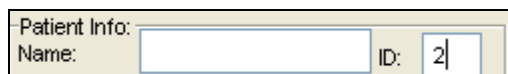
☐ Examination Confirmed

Doctors Notes....

Figure 13: Doctor Confirmation Screen.

6.2 Searching the patient number.

To enable a doctor to confirm an examination, it is required that he/she enters the patient id which they are provided in an e-mail in the Patient “ID” box in the top left of the interface under “Patient Info” as seen below in figure 13.1:

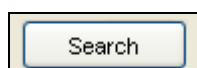


Patient Info:

Name: ID:

Figure 13.1: ID entry

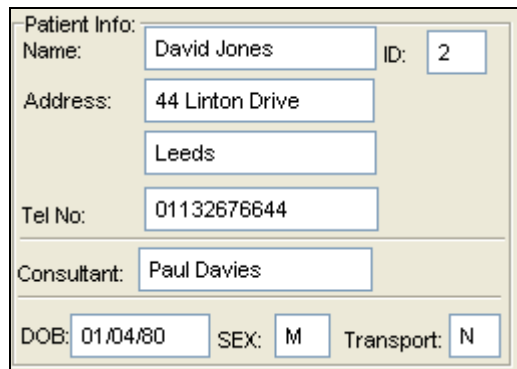
The doctor then needs to search for the Patient by clicking the “Search” button at the bottom of the interface as seen in Figure 13.2 below:



Search

Figure 13.2: Search Button

Through searching, this will then bring up the patient information relating to that particular unique patient ID number, seen in Figure 13.3 below, furthermore, the interface will also show the related examination and clinical information about that patient.

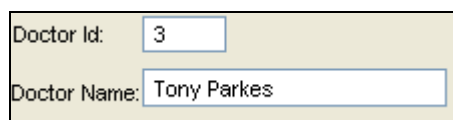
A form titled "Patient Info:" containing several input fields. The "Name:" field contains "David Jones" and the "ID:" field contains "2". The "Address:" field contains "44 Linton Drive" and "Leeds". The "Tel No:" field contains "01132676644". The "Consultant:" field contains "Paul Davies". The "DOB:" field contains "01/04/80", the "SEX:" field contains "M", and the "Transport:" field contains "N".

Patient Info:	
Name:	David Jones ID: 2
Address:	44 Linton Drive Leeds
Tel No:	01132676644
Consultant:	Paul Davies
DOB:	01/04/80
SEX:	M
Transport:	N

Figure 13.3: Patient Info

6.3 Entering unique doctor id.

The next step in a doctor confirming an examination is firstly to enter details about the doctor who is confirming the exam; by entering the Doctor ID in the doctor ID box, as shown in the figure below, will automatically bring up the name of that doctor in the doctor name box below it:

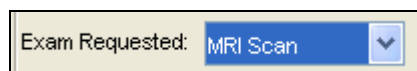
A form with two fields. The "Doctor Id:" field contains "3" and the "Doctor Name:" field contains "Tony Parkes".

Doctor Id:	3
Doctor Name:	Tony Parkes

Figure 13.4

6.4 Changing Suggested Exam.

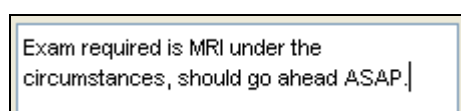
The next step a doctor needs to take is to change or confirm the suggested examination, in the examinations details section, the figure below shows the doctor selecting an MRI Scan for the patient.

A form with a single field labeled "Exam Requested:" containing a dropdown menu with "MRI Scan" selected.

Exam Requested:	MRI Scan
-----------------	----------

6.5 Entering any concluding notes.

The doctor can enter some concluding notes explaining why an examination was not confirmed, why it should or should not go ahead and in some cases the time scale and the reasoning behind it, Figure 13.6 below shows this:

A form with a single text area containing the text "Exam required is MRI under the circumstances, should go ahead ASAP.".

Exam required is MRI under the circumstances, should go ahead ASAP.

Figure 13.6: Concluding Doctors Notes

6.6 Confirming the exam and saving.

The final step is then to confirm the examination, this is a two step procedure; first the exam date must be specified in the correct format of DD/MM/YY and then clicking the “examination confirmed” check box – to indicate closure for this patient. The figure 13.7 shows these actions:

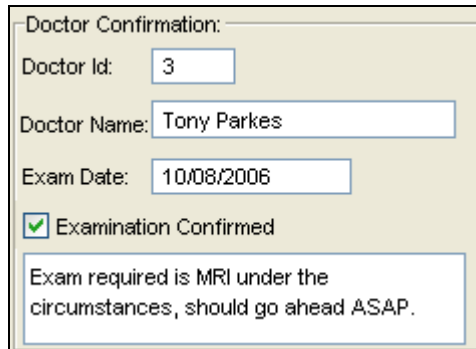
A screenshot of a web form titled "Doctor Confirmation:". It contains several input fields: "Doctor Id:" with the value "3", "Doctor Name:" with the value "Tony Parkes", and "Exam Date:" with the value "10/08/2006". Below these fields is a checked checkbox labeled "Examination Confirmed". At the bottom of the form, there is a text box containing the message: "Exam required is MRI under the circumstances, should go ahead ASAP."

Figure 13.7: Confirming the examination.

Once the examination has been confirmed the changes then needed to be saved into the database, figure 13.8, the save will update all the information in the database relating to the patient number 2.

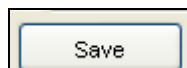
A screenshot of a single button labeled "Save".

Figure 13.8: Save Button.

Chapter 7

Saving and Searching the Database

7.1 Saving.

Although saving to the database has been detailed previously within the user-guide, it is important to understand why it is so imperative and how it works to ensure no errors are experienced. When clicking on the save button it is essential that all information has been correctly entered, errors may be experienced if the user, for example, omits anything related to the patient such as his or her sex, d.o.b. this is because all patient information is mandatory and the database which the data is saved into does not accept “NULL” or empty patient attributes; this is one reason for attaining an SQL Error. Other errors which can be experienced ‘violation’ errors, this is where data is attempting to be entered into the database which is violating the database rules set-up, for example, the database will only accept a person’s gender in the format of ‘M’ or ‘F’ if a user enters anything other than these two values a violation will occur upon saving. The only other errors may occur if the dates are not in the format of DD/MM/YY, as this is the format which will be accepted in the database.

7.2 Searching.

Searching is a mandatory first action which all doctors who have been instructed to confirm an examination must perform. There is one error which may occur when searching and that is if a patient id is not found. If the error message “patient details not found” appears, this means either the ID has been wrongly keyed or the patient does not exist in the database. It is important that the doctor checks the ID which they have keyed and re-try.

Chapter 8

Management

8.1 System Extension

The current system is a version 1.0 build, prototype. It is currently set up to a *dummy* MSSQL Server database which takes the data-types and names from the departments existing SQL Server database, therefore allowing less implementation when integrating the system. Obviously, there are a number of extensions, which will be needed to the second system build to turn it into a fully functional system, which can actually be used within daily practices. The extensions are listed below:

System Extensions:

- Modify coding to connect correctly to the existing departmental patient database.
- Create/Modify a new/existing table to incorporate the saving and retrieval of examination details.
- The Clinical Help System needs to include further information to support more queries from users.
- Examination Scheduling could be included, this could be an extra tab added to the main interface to organise the scheduling of examinations; this may also be password protected to allow only authorised personnel.

8.2 Code Modifying.

To change the connection to the database it is advised that the Eclipse development environment is used as this will provide a environment to modify the code and will highlight any syntactical errors which may prevent the code from compiling. There are a number of classes which were created relating to different parts of the system, fully commented to make editing easier for future developers; the class which is of importance to modify the database code is the SQLExecutor class. Firstly the string URL will need to be changed to include the path of the sever, leave the default port as 1433, which is the default for SQL Server, then change the name of the database.

```
private String url = ("jdbc:jtds:sqlserver://csms11.leeds.ac.uk:1433/fyp_scs3sw");
```

Secondly, the username and password will need to be changed which has been assigned to the database. This code is directly below the URL string which is passed, this can be seen below:

```
private String username = "*****";  
private String password = "*****";
```

The modifying here is relatively straight forward, the username should be where the stars are after the equals and the password where the stars are after the password equals.

It will also be required to set up a new table for the examination details, if this does not already exist; the SQL to perform this is listed below:

```
CREATE TABLE exam_tbl (exam_id integer primary key,  
    Dept_req varchar(20) NOT NULL,  
    Ref_category varchar(16) NOT NULL,  
    Urgency varchar(12) NOT NULL,  
    Status varchar(10) NOT NULL,  
    Appt_date datetime NOT NULL,  
    Clinical_info varchar(100),  
    Diabetic_info varchar(1) NOT NULL,  
    CONSTRAINT check_diabetic CHECK (Diabetic_info = 'Y' OR Diabetic_info = 'N'),  
    Exam_required varchar(15) NOT NULL,  
    Date_of_exam datetime,  
    Extra_info varchar(100),  
    Allergies varchar(1) NOT NULL,  
    CONSTRAINT check_allergies CHECK (Allergies = 'Y' OR Allergies = 'N'),  
    Pregnancy varchar(1) NOT NULL,  
    CONSTRAINT check_preg CHECK (Pregnancy = 'Y' OR Pregnancy = 'N'),  
    Menstrual varchar(3) NOT NULL,  
    CONSTRAINT check_men CHECK (Menstrual = 'Y' OR Menstrual = 'N' OR Menstrual = "N/A"),  
    doctor_id integer NOT NULL references dbo.doctor_tbl(doctor_id),  
    patient_id integer NOT NULL references dbo.patient_tbl(patient_id));
```

8.3 Drivers.

It may be that new drivers might be required or preferred to the ones currently used. Therefore it is important to know their role in helping to connect to the database. JDBC technology-enabled drivers are available from a number of vendors; they turn calls within the interface into database readable calls and hence that is why they are required for storing data and retrieving data. To install a new driver, you first must download the zip file or executable from the relevant vendor, and you need to ensure that this file is called within your classpath. All one needs to know then is the name of Driver and the DataSource implementations and the and you're all set! To set the classpath depends entirely on the driver being used, with jTDS the drivers come within a jar file, it is important that that the instructions are read on the vendors website on how to put the file into the classpath, once this is done two sections of code will need to be edited; I have anticipated that this may need to be changed and have commented the code within the SQLExecutor class which will show where the Driver and DataSource lines will need modifying.

8.4 Recommendations.

The following recommendations are for Build 1.1 and were unable to be corrected and changed within the current system build.

Build 1.1 (Recommendations only):

- Display all the dates as 00/00/00 and when clicked it disappears.
- 'AM/Pm' should be made a drop down hard-coded selectable box, as too 'Sex' and 'Transport'.
- Clinical Information box should include breaks when text reaches the end of the line, it currently all goes on one line requiring the user to press enter at end of line.
- More clinical information included for users of the system to help in examination selection.
- Consultant could possibly be identified by his or her id number instead of name as there could be a consultant with two names.
- Inclusion of a doctors bleep number or extension for contact.
- Extra room for another contact telephone number.

Appendix G

Usability Testing

Usability Questionnaire (March 2006)

Examination Request System – Usability Study.

I would like to thank all participants for taking time to complete this study, the purpose of the study is to gain a greater understanding of any problems which a user may experience when using the Examination Request system. The Examination Request system builds upon the manual paper based system for the requesting of examinations electronically.

The following sections have been prepared for your completion, in order to gauge the reliability of the system, it is important that you, as a user, complete all tasks in each section of the study and answer questions *honestly*, it is important to stress that all stages must be completed so the results can be collated and analysed properly.

The study will begin with some personal questions with regard to you as a user; this is to enable a profile to be built about the different types of users.

1. Age: 16-20 ☐ 21-29 ☐ 30-40 ☐ 41-55 ☐ 56 + ☐

2. Sex: Male ☐ Female ☐

3. Occupation:

4. How would you rate your expertise? Novice ☐ Intermediate ☐ Expert ☐

5. Have you used the previous system? Yes ☐ No ☐

6. (If yes to 5) How easy did you find it? 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. (If yes to 5) How efficient did you find it... 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

In terms of accuracy & speed?

If you are a user of the old system, please take time to fill in the questions at the end of this Questionnaire so the results can be compared.

Rating Legend: Ratings are in the range of 1 – 5 and correspond to:

1 = "Very Difficult" 2 = "Difficult" 3 = "Moderate" 4 = "Easy" 5 = "Very Easy"

Furthermore – some questions may require a “Yes” or “No” answer.

Examination Request System

Task One: Interaction with the log-in screen.

Bring up the log in screen.

Bring up the log in screen					
(a) Enter your assigned username	1	2	3	4	5

(b) Enter your assigned password	1	2	3	4	5
(c) Attempt to log-in	1	2	3	4	5

Task Two: Interaction with Examination Request Interface.

Spend a few of minutes acclimatising to the main interface, and then perform the tasks relating to Entering patient information below.

(a) Enter some personal details for a patient	1	2	3	4	5
(b) Does the interface provide easy entry for data?	Yes / No				
(c) Enter whether the patient requires transport	1	3	3	4	5
(d) Was there enough information for this?	Yes / No				
(e) Specify the dept. required, reference category, urgency and patient status.	1	2	3	4	5
(f) Were there enough options to do this?	Yes / No				
(g) Enter an appointment date and time	1	2	3	4	5
(h) Did you know the format to enter this?	Yes / No				

Task Three: *More* Interaction with Examination Request Interface – Clinical Info

Please perform the specified tasks relating to entering and specifying the patient clinical information.

(a) Enter some key clinical info about a patient	1	2	3	4	5
(b) Was there enough room to enter this?	Yes / No				
(c) Specify whether the patient is diabetic	1	2	3	4	5
(d) Specify whether the patient may be pregnant	1	2	3	4	5
(e) Specify whether the examination should or should not go ahead if period is over-due	1	2	3	4	5
(f) Find and open the clinical help	1	2	3	4	5
(g) Do you know what this screen is asking you to do?	Yes / No				
(h) Start at the root and follow the sequence according to the clinical info you entered	1	2	3	4	5
(i) Were there enough choices	Yes / No				
(j) Were you able to get a suggestion from the support system?	Yes / No				

Please can you briefly outline below whether or not you think the clinical support is beneficial to aid in selecting an examination.

.....
.....
.....

Task Four: *More* Interaction with Examination Request Interface – Exam Details

Please perform the specified tasks relating to entering and specifying examination information about a patient.

(a) Locate the Examination Details entry section	1	2	3	4	5
(b) Specify whether patient has had an exam	1	2	3	4	5
(c) Enter the date of previous exam if applies	1	2	3	4	5
(d) Did you know the required format?	Yes / No				
(e) Enter the place of previous exam if applies	1	2	3	4	5
(f) Did you know what was required for entry?	Yes / No				
(g) Specify whether the patient is allergic	1	2	3	4	5
(h) Do you understand why (g) is important?	Yes / No				

- | | | | | | |
|--|----------|---|---|---|---|
| (i) Select an examination | 1 | 2 | 3 | 4 | 5 |
| (j) Specify whether you want doctor confirmation | 1 | 2 | 3 | 4 | 5 |
| (k) Save all your progress so far | 1 | 2 | 3 | 4 | 5 |
| (l) Were there any errors in saving? | Yes / No | | | | |
| (m) If so what was the error message? | | | | | |
| (n) do you know how to resolve this? | Yes / No | | | | |
| (o) If you answered no, why? | | | | | |

Task Five: More Interaction with Examination Request Interface – Doctor Request

Please perform the specified tasks relating to confirming an examination from the point of view of a doctor.

- | | | | | | |
|---|----------|---|---|---|---|
| (a) Enter the patient id you entered previously | 1 | 2 | 3 | 4 | 5 |
| (b) Search for the patient using search | 1 | 2 | 3 | 4 | 5 |
| (c) Did you know straight away how to do this? | Yes / No | | | | |
| (d) Did the patient details display properly? | Yes / No | | | | |
| (e) Enter doctor id and name | 1 | 2 | 3 | 4 | 5 |
| (f) Change the “Exam requested” | 1 | 2 | 3 | 4 | 5 |
| (g) Was it obvious what you needed to do? | Yes / No | | | | |
| (h) Enter the date of exam | 1 | 2 | 3 | 4 | 5 |
| (i) Confirm the exam | 1 | 3 | 3 | 4 | 5 |
| (j) Enter some concluding notes | 1 | 2 | 3 | 4 | 5 |
| (k) Save the changes | 1 | 2 | 3 | 4 | 5 |
| (l) Were there any errors in saving? | Yes / No | | | | |
| (m) If so what was the error message? | | | | | |
| (n) Do you know how to resolve this? | Yes / No | | | | |
| (o) If you answered no, why? | | | | | |

Were there any errors when using the Examination Request System?

.....

What, if at all, did you find confusing or difficult about your experience with the Examination Request System?

.....

How would you improve the system if you could?

.....

Comparison of the new electronic Examination System and the manual system

If you have previously used the old, manual system then please complete a couple of questions below to help gauge any improvements or discrepancies with new system.

How did you find the manual system in terms of ease of use?

Easy ☐ Moderate ☐ Difficult ☐

How did you find the electronic Examination System in terms of ease of use?

Easy ☐ Moderate ☐ Difficult ☐

What was the likelihood with making a mistake(s) with the manual system?

Little ☐ Moderate ☐ High ☐

What was the likelihood with making a mistake(s) with the electronic system?

Little ☐ Moderate ☐ High ☐

How would you rate the information quality of the manual system?

Poor ☐ Moderate ☐ High ☐

How would you rate the information quality of the electronic system?

Poor ☐ Moderate ☐ High ☐

Thank you for participating in this study for the Examination Request System

Usability Test Results

Users:	1	2	3	4	5	Average
Personal Information						
Sex	F	F	M	M	M	2F 4M
Age	41-55	21-29	30-40	41-55	41-55	N/A
Occupation	Nurse	Nurse	Radio.	Doctr.	Doctr.	N/A
Expertise	Novice	Interm.	Interm.	Interm.	Novice.	2Nov – 3 Interm.
Previous Usage of system						
Used pervious?	Yes	Yes	Yes	Yes	Yes	5Y 0N
How easy was it	3	3	2	4	4	3.2
How efficient	3	3	3	3	3	3
Task 1: Interaction with log-in						
Username	5	5	5	5	5	5
Password	5	5	5	5	5	5
Log-in process	5	5	5	5	5	5
Task 2: Exam Request Interface						
Patient data	4	5	4	4	4	4.2
Easy Entry?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Transport	5	4	5	5	5	4.8
Enough info?	Yes	Yes	Yes	Yes	Yes	5Y 0N
More patient info	4	5	5	5	4	4.6
Enough options?	Yes	Yes	Yes	No	Yes	4Y 0N
Appt. time/date	2	3	4	4	3	3.2
Format known?	No	No	Yes	Yes	No	3Y 2N
Task 3: Clinical Info						
Clinical entry	4	4	5	4	5	4.4
Enough room?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Diabetic	5	5	5	5	5	5
Pregnancy	5	5	5	5	5	5
Cont. if o/due?	4	4	5	4	5	4.4
Find clinical help	5	5	5	5	5	5
Know what it is?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Starting at root	3	4	4	4	5	4
Enough choices	Yes	Yes	Yes	No	Yes	4Y 1N
Get a suggestion?	Yes	Yes	Yes	No	Yes	4Y 1N
Task 4: Exam Details						

Exam details	4	4	5	4	4	4.2
Patient had exam?	4	4	5	3	5	4.2
Date of exam?	4	4	5	4	4	4.2
Know format?	No	No	No	No	No	0Y 5N
Place?	3	4	3	4	4	3.6
Know what to do?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Allergies	4	4	5	5	5	4.6
Understand import	Yes	Yes	Yes	Yes	Yes	5Y 0N
Select Exam	5	5	5	5	5	5
Doc conf?	5	5	5	5	5	5
Save progress	5	5	5	5	5	5
Errors	Yes	Yes	No	No	Yes	3Y 2N
Know resolution?	Yes	Yes	N/A	Yes	No	3Y 1N

Task 5: Doctor request

Enter patient id	5	5	5	5	4	4.8
Search patient	5	5	5	5	5	5
Know how?	No	Yes	Yes	Yes	Yes	4Y 1N
Display prop?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Enter doc id/name	3	2	4	5	5	3.8
Change exam	3	4	4	4	4	3.8
Obvious?	Yes	Yes	Yes	Yes	Yes	5Y 0N
Date Entry	3	3	4	3	4	3.4
Confirm exam	5	5	5	5	5	5
Concluding notes	5	5	5	5	5	5
Save Changes	5	4	5	5	4	4.6
Errors?	Yes	No	No	No	Yes	2Y 3N
Resolve?	Yes	N/A	N/A	N/A	Yes	2Y 0N

Comparison of old system to new one

Easiness (old)	Moderate	Moderate	Moderate	Moderate	Moderate	5 Moderate
Easiness (new)	Moderate	Easy	Easy	Easy	Moderate	3 Easy 2 Moderate
Mistakes (old)	Moderate	Moderate	High	High	Moderate	3Mod 2 High
Mistakes (new)	Moderate	Little	Little	Moderate	Little	3 Little 2 Moderate
TOTAL AVERAGES	4.2	4.2	4.6	4.5	4.3	4.4

Summary of Results:

The table on the previous page provided a break down of all the results given from the 5 users of the system. This summary details some of the written answers provided and interpreted into a list of system pros and cons, overall the feedback was generally that the system was very good and beneficial, however there were a few points raised to further improve the system.

System Pros:

- ▶ The ability to search for patients was very easy.
- ▶ Use of drop down boxes prevented keying errors / fast.
- ▶ Clinical help was seen as innovative and helpful
- ▶ The e-mail system was seen as a quick way of contacting the doctor
- ▶ Clinical help assisted well in examination suggestion.
- ▶ The layout was much improved.
- ▶ Error messages proved helpful in fixing a save problem
- ▶ Tool tips were useful to find the correct format
- ▶ Lots of room for clinical information
- ▶ Doctors notes aided in giving reason for confirming an examination

System Cons:

- ▶ There could be more options added to the clinical support system.
- ▶ Unclear on the format of the dates
- ▶ Would be clearer to indicate the date format above the entry
- ▶ Entering the doctor name caused some confusion on what was expected
- ▶ Errors when attempting to save – wasted time
- ▶ The ability to save patients
- ▶ “Sex” and “Transport” could have been hard coded in a drop-down box
- ▶ The “place of examination” was unclear what was required
- ▶ Separate screen for the doctor confirmation

Appendix H

Heuristic Evaluation

Examination Request System Heuristic Evaluation

Participants are required to evaluate as an expert evaluator, following a set of heuristic principles adopted for the use of the Examination Request System, please use the ratings below to decide upon the relevant criteria for each heuristic you think has been satisfied.

Scenario:

You are a nurse needing to use the Examination Request System, you are under intense pressure as a patient is unwell, you need to conduct the request as quickly as possible, a doctor is standing beside you waiting to confirm the examination, once it has been determined, both you nor the doctor have used the interface before.

Two Tasks:

- 1: [Familiarisation] Attempt to log-into the system, quickly enter the patient details, enter all the related clinical information and then select and examination and save to the database.
- 2: Log-in again this time taking your time, pay attention to detail and follow the heuristics below. After considering the criteria, make note of the ranting which you think applies.

Heuristic	Heuristic Criteria	Ratings 1 = very poor, 2 = poor, 3 = average, 4 = good, 5 = very good
Visibility	<ol style="list-style-type: none">1. Interface should show the relevant information to the user to be able to complete each task.2. Highlighting should be used to good effect.3. Response time should be short between user performing a task and the one being executed.4. The interface should flow well. <p><i>Feedback:</i> Use of highlighting is small, information flows from left to right nicely, response time and execution time is short.</p>	<div>4</div> <div>2</div> <div>4</div> <div>4</div>

Language	1. Language should be fully understandable to users. 2. Technical terms should be avoided or explained. 3. Technical computing abbreviations should not be used. 4. User input fields should not be restrictive. <i>Feedback:</i> Language is fairly simplistic although could be simpler, there are a few abbreviations which can cause some confusion initially.	3 4 3 4
User Control	1. User should be able to exit at any time. 2. User should be able to reverse an action. 3. User should be able to return to previous screens to make changes to choices. 4. Short-cuts for more advanced users. <i>Feedback:</i> No exit for users although can click on the exit cross, no short cuts apart from tab	3 4 4 2
Consistency and Standards	1. Tasks, which are similar, should be performed in the same manner. 2. Similar layouts for forms and menus throughout 3. Fonts and colours consistent throughout. 4. Information on how to complete a task should be on same screen as the task. <i>Feedback:</i> layout is excellent and neatly presented, small use of colour not overly used, information is related to a task	5 5 4 4
Recognise Errors, Diagnose them and Resolve.	1. Users should always be given feedback 2. Feedback should be given if an error occurs 3. The system should always respond so it doesn't appear that the system has crashed, 4. Error messages should be phrased in a way that errors can be resolved. 5. Error messages should be concise and to the point. <i>Feedback:</i> System responds however its sometimes hard to tell if it has crashes, error messages are excellent and informative	3 4 3 4 5
Prevention of Errors	1. Users should be able to select from drop-down boxes than having to key information all the time. 2. Users should be able to confirm if they want to exit rather than exit immediately. 3. System should not present two similar commands to the user. <i>Feedback:</i> Drop down boxes effectively used to capture repeated information, no exit confirmation which is poor, there are some similar commands presented to the user.	4 2 3

Recognition opposed to recall	1. Users should not be asked to remember info, only use it when needed. 2. Users should have an easy way of identifying patients. 3. Users should be told date formats. 4. Only a few simple rules should be known to complete each task. <i>Feedback:</i> One date format where the user isn't told, information is easy to understand.	4 5 3 4
Aesthetic and minimalist design	1. User shouldn't be overloaded with information. 2. Colour should be used effectively. 3. System should be able to be usable for all types of people including disabilities <i>Feedback:</i> Using the tab no mouse is needed which is good as interface flows from left to right, effective use of colour – nothing unreadable.	3 4 3
Help information and documentation	1. Details should be provided to users for appropriate system use. 2. Manual – task oriented. 3. Manual should not be over-long and should be accessible for skim reading main points. 4. Manual should be easy to look-up <i>Feedback:</i> Excellent documentation which details the tasks in a sequential manner, look-up is relatively easy.	5 5 4 4
	Average ratings of the 3 scores:	131/175 (74%)

Appendix I

Screen Sources

Main Examination Request Screen, with patient details filled in for a patient:

<http://img89.imageshack.us/img89/4041/examscreen1aq.jpg>

Main Examination Request Screen, with clinical information entered for the patient:

<http://img105.imageshack.us/img105/7421/examscreen27qi.jpg>

Clinical Support Screen which uses the clinical information to derive an examination suggestion:

<http://img98.imageshack.us/img98/2949/clinicalsupport3lzy.jpg>

Main Examination Request Screen, with Exam Details filled in – requesting a doctor to confirm:

<http://img59.imageshack.us/img59/7426/examscreen35sa.jpg>

Email which got sent as a result of checking the Doctor Confirmation box:

<http://img223.imageshack.us/img223/4716/emailconfirm7kn.jpg>

Main Examination Request Screen after a doctor has searched the patient number and then confirmed the correct examination for the patient:

<http://img223.imageshack.us/img223/8642/examscreen40ri.jpg>