Table of Figures

Index of Tables

Chapter 1: Introduction

In order for any computer system to be effective, it must allow intended users to accomplish their tasks in the best way possible. This concept of usability, however, is often little more than an afterthought as software developers focus on other aspects of system development such as product functionality and performance. It is the author's experience that in St John of God Services, resource restrictions, looming deadlines and a general lack of expertise among the software development team regarding the concept of usability are just some of the reasons why software systems and components, when implemented, have a less than high level of usability.

Usability is important for many different reasons. From the user's perspective it can mean the difference between performing a task accurately and completely or not, and the process being enjoyable or frustrating. For software developers, usability is important because it can mean the difference between the success or failure of a system. From a management point of view, software with poor usability can reduce the productivity of the workforce to a level of performance worse than without the system. In all cases, lack of usability can cost time and effort, and can have a significant influence on the success or failure of a system. (usabilityfirst.com)

In order to promote consistency in user interface development and improve the usability of software systems developed by St John of God Services, a formal usability evaluation strategy is sought which can be implemented as part of the iterative software design process.

Before commencing research, an overview of St John of God Services in Ireland is offered showing the different activities conducted by the organisation in their various mental health and intellectual disability services throughout Ireland. In this chapter, particular attention is paid to the structure and work practices of the software development department. A profile of the staff within the department is then included as is an example depicting the software development process (from requirements to implementation) for a recently developed software component. This background is necessary as it influences the selection of the most appropriate usability evaluation technique for use in St John of God Services.

A literature review is carried out in two sections. Chapter 3 discusses the concepts of usability and of evaluation and shows why many organisations neglect to include usability evaluation as part of their software design process. There are many benefits which may accrue to an organisation from the introduction of usability evaluation methods and these are demonstrated in this chapter.

Chapter 4 forms the second part of the literature review and entails a detailed appraisal of some of the most popular usability evaluation techniques being used by healthcare organisations today. Divided into descriptive and predictive strands, the many and varied approaches to usability evaluation are reviewed and their relative suitability to the St John of God case assessed given the current work practices and resource constraints outlined in chapter two.

Chapter 5 commences with a presentation of the goals which it is hoped will be achieved by the introduction of a usability evaluation methodology in St John of God Services. Following this, there is a discussion of the various factors which might influence the selection of a usability evaluation technique. The relevance of each of these factors to the St John of God case is also discussed so that the most appropriate technique can be selected. The chapter concludes with the selection of the usability evaluation technique deemed most suitable and the decision is taken to use Nielsen's Heuristic Evaluation.

In Chapter 6, this chosen technique is tested against a recently developed software module to establish its effectiveness as a means of identifying usability issues. A list of tasks are drawn up and a software developer, with no previous involvement in the component's development, acting as the evaluator, is asked to inspect the various dialog elements and to compare them with a list of recognised usability principles – the heuristics.

Finally, conclusions of the dissertation are discussed and suggestions are offered as to further work which would compliment this study.

Chapter 2: St John of God Services in Ireland

2.1 Introduction

Before we begin the process of choosing a usability evaluation strategy for software development in St John of God Services, an overview of St John of God Services as an organisation in Ireland is provided, paying particular attention to the ICT function which is based in Stillorgan Co Dublin. A brief introduction to the two main in-house developed systems, the Mental Health Information System (MHIS) and the Intellectual Disability Information System (IDIS) will be followed by a profile of the current software development team as well as a profile of the users of both systems. In order to illustrate the current work practice regarding software development in SJOG, the process of developing a new bed management component for the IDIS will be laid out by way of example. A good understanding of the working environment both in terms of resources, users and current work practices is important as these influence which usability evaluation technique will to select.

2.2 St John of God Hospitaller Services

St John of God Hospitaller Services is an international healthcare organisation, run by St John of God Brothers with over 250 hospitals and centres throughout the world. The Organisation is run by St John In Ireland, St John of God Hospitaller Services (SJOG) provides mental health services, care for older people and services for children and adults with intellectual disabilities. Each year over 3000 individuals receive support through mental health and intellectual disability services operated by over 2000 staff and volunteers including 36 St John of God Brothers.

Below is a table detailing the work carried out by each of our mental health and intellectual disability services in Ireland.

Mental Health	
St John of God Hospital, Stillorgan, Co Dublin	St. John of God Hospital is an acute psychiatric teaching hospital with 210 beds, out-patient and day hospital services. The hospital accepts patients from all over the country. Patients are also referred through the community mental health service for southeast Dublin run by St. John of God Services and based at Cluain Mhuire, Blackrock.
Cluain Mhuire Community, Mental Health Services Blackrock, Co Dublin	Provides community mental health programmes for the people of Dun Laoghaire-Rathdown.
Lucena Services, Rathgar, Dublin 6	Provides child and adolescent mental health programmes in south Dublin city and county and in county Wicklow.
Granada Institute, Shankhill, Co Dublin	Contributes to the safety, protection and well-being of children, adolescents and vulnerable adults through the provision of assessment and treatment services for those who have experienced sexual abuse and for those who have perpetrated sexual abuse and for the families of both.

Table 1 - Mental health services in St John of God Services (Ireland)
Source : http://www.sjog.ie

Intellectual Disability	
St Augustine’s School, Blackrock, Co Dublin	Co-educational day and boarding school providing educational, social, recreational and vocational training programmes for 160 students with special education needs.
Kildare Services, Celbridge, Co Kildare	Provides a network of day, training, employment services and residential services, as well as a respite service to over 320 children and adults with intellectual disabilities.
North East Services, Dunleer, Co Louth	Provides 630 children and adults with residential and day services in counties Louth, Monaghan and Meath.
Callan Institute, Shankhill, Co Dublin	Provides consultation and training in the use of effective, non-aversive methods for supporting people who have challenging behaviour.
Carmona Services, Dun Laoghaire, Co Dublin	Provides a network of residential and day services in southeast Dublin and north County Wicklow for 260 children and adults.
Brennán Services, Tralee, Co Kerry	Provides training, employment, social and residential programmes for people with intellectual disability in County Kerry.
Menni Services, Island Bridge, Dublin 8	Supports 425 children and adults attending a network of residential and day programmes. The catchment area is south west Dublin including Tallaght.
STEP Enterprises	Provides work and supported employment for 140 people in south county Dublin.
Northern Ireland	Provides residential and independent living services for adults with an intellectual disability and residential care for frail, elderly people.

Table 2 - Intellectual disability services in St John of God Services (Ireland)
Source : http://www.sjog.ie

2.21 Provincial Administration

Located on the grounds of St John of God Hospital in Stillorgan, Provincial Administration is the administrative head quarters for St John of God Services in Ireland. Here, the functions of Human Resources, Research, Programme Development, Finance, Library Services, Publications and Information and Communication Technology (ICT) are centred.

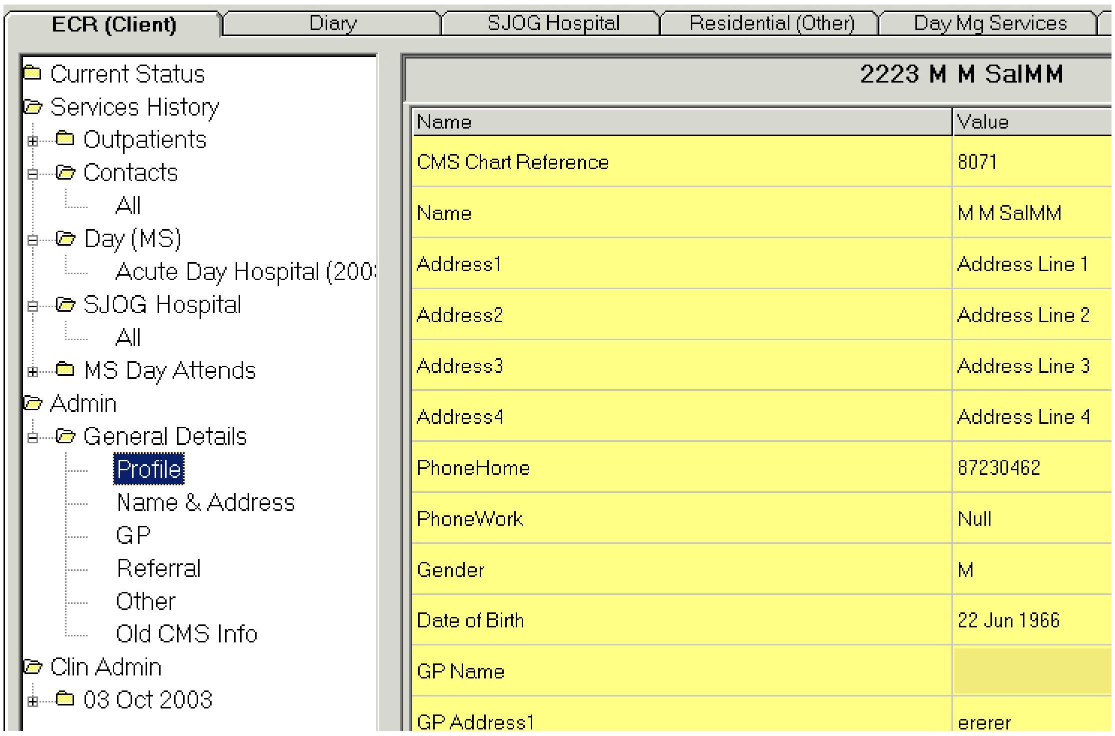
2.3 Software Services

The ICT Department is comprised of Infrastructure and Software Services divisions. Software systems relating to mental health and intellectual disability are generally developed in-house but where resource constraints dictate, external contractors are occasionally employed to develop new components of existing systems.

The two main in-house developed systems currently in place in SJOG in Ireland are the Mental Health Information System (MHIS) and the Intellectual Disability Information System (IDIS). Both systems which are developed using Visual Basic 6 with data held in a SQL Server 7 database.

2.31 Mental Health Information System (MHIS)

Previously a COBOL based system, the current Windows version of the MHIS has been in place for 10 years. Initially used as a patient administration system, it has evolved significantly since its inception and now encompasses an electronic patient record where a diversity of clinical data is held for each inpatient and outpatient. From consultant to social worker, all interactions with patients are recorded electronically as well as other patient related data such as laboratory results, discharge summaries, alerts, referrals and medication. The system is constantly being enhanced as requested by clinicians. As can be seen from the screen shots in figure 1 and figure 2, the layout of the main user interface is similar for both the MHIS and IDIS systems.



Name	Value
CMS Chart Reference	8071
Name	M M SalMM
Address1	Address Line 1
Address2	Address Line 2
Address3	Address Line 3
Address4	Address Line 4
PhoneHome	87230462
PhoneWork	Null
Gender	M
Date of Birth	22 Jun 1966
GP Name	
GP Address1	ererer

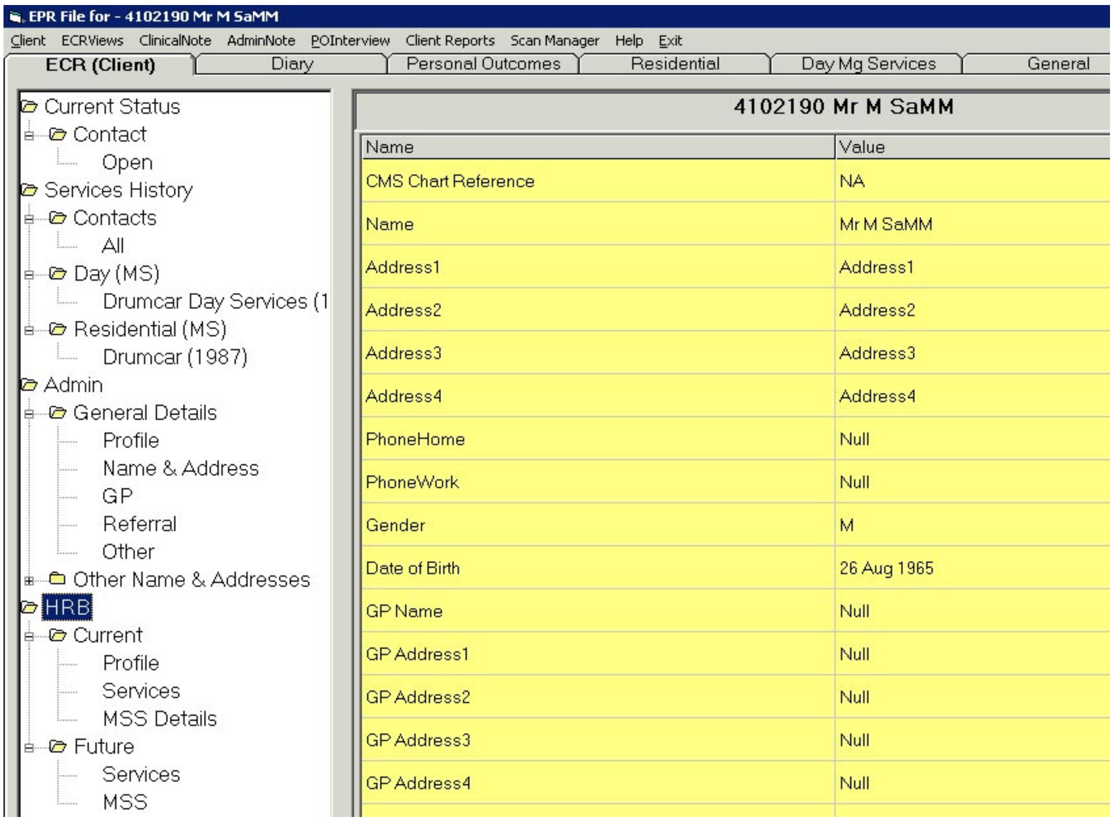
Fig 1 – MHIS main screen

All information about a particular patient can be accessed through the above screen but in order to add to a patient's record, components are launched through which data can be added to a patient's record.

2.32 Intellectual Disability Information System (IDIS)

Development of version 1 of the IDIS has recently been completed and the system is currently being implemented in each of our intellectual disability services. Using the same database and front end design as the MHIS, the IDIS system stores both administrative data and data relating to the day and residential services a client receives from SJOG. Since the completion of phase 1 of development, work has commenced on a number of software components which will improve the value of the system for those who use it. The components currently in development are:

- Reporting Module
    - Allows users to create and share reports on client data
- HRB Upload Module
    - Facilitates nightly upload of client data from IDIS system to Health Research Board.
- Bed Management
    - Allows users to monitor bed occupancy in their service through a graphical interface.


EPR File for - 4102190 Mr M SaMM
Client ECRViews ClinicalNote AdminNote POInterview Client Reports Scan Manager Help Exit
ECR (Client) Diary Personal Outcomes Residential Day Mg Services General

- Current Status
    - Contact
        - Open
- Services History
    - Contacts
        - All
    - Day (MS)
        - Drumcar Day Services (1
    - Residential (MS)
        - Drumcar (1987)
- Admin
    - General Details
        - Profile
        - Name & Address
        - GP
        - Referral
        - Other
    - Other Name & Addresses
- HRB
    - Current
        - Profile
        - Services
        - MSS Details
    - Future
        - Services
        - MSS

4102190 Mr M SaMM

Name	Value
CMS Chart Reference	NA
Name	Mr M SaMM
Address1	Address1
Address2	Address2
Address3	Address3
Address4	Address4
PhoneHome	Null
PhoneWork	Null
Gender	M
Date of Birth	26 Aug 1965
GP Name	Null
GP Address1	Null
GP Address2	Null
GP Address3	Null
GP Address4	Null

Fig. 2 – IDIS main screen

All information about a particular client can be accessed through the screen above but as with the MHIS, certain tasks require new components to be launched. For example, where a client has been moved from one bed to another bed or where they have been transferred to a hospital for treatment, the Bed Management component is launched allowing the move to be recorded and the clients bed history to be maintained. A more detailed account of the Bed Management component will follow.

2.33.1 Software development team

Within the Software Services group there are five full time staff whose roles are summarised as follows

2.331 IT Manager

The IT manager oversees the ICT department and with a software engineering background carries out occasional software development work and oversees all software development projects developed at St John of God services.

2.332 Database Administrator

Carries out all tasks relating to the administration of several test and live databases. The database administrator also provides technical support for the MHIS and IDIS systems and is involved in testing of new systems and components.

2.333 Business Analyst

Oversees the process of acquiring and integrating bespoke systems and of several externally developed software systems. The business analyst also carries out testing of new systems and components and occasional software development work.

2.334 Software Developer x 2

Duties include maintenance of in-house developed software systems as well as development of new systems and new components for existing systems. Involved at all stages of the Software development lifecycle. The developers are also responsible for the creation of any new databases and amendments to the structure of existing databases.

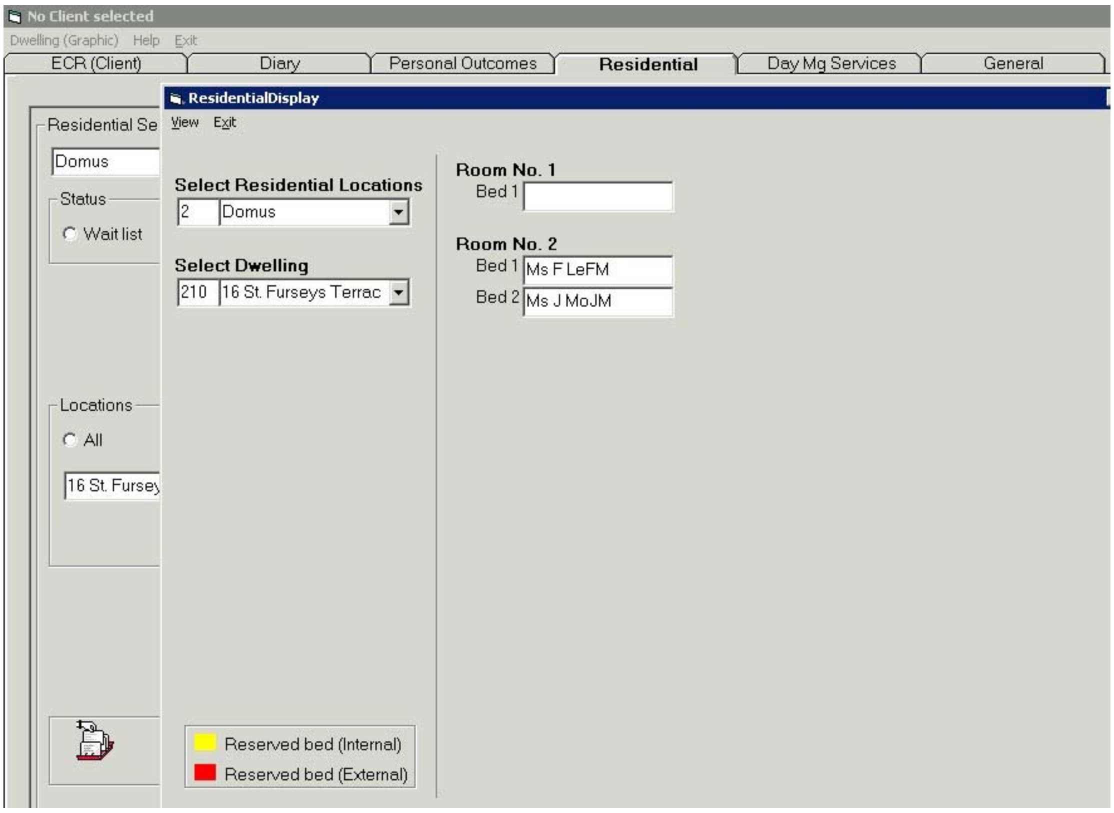
2.4 User profile.

Users of the MHIS and IDIS systems are generally clinical staff who have limited IT experience and knowledge although staff involved in mental health, tend to be more IT literate than their colleagues in intellectual disability as the MHIS has been in place for 10 years. It is the author’s experience that these staff members, whether involved in mental health or intellectual disability, often have a negative attitude towards existing systems and show little motivation to use new systems. New systems are viewed as “just more work” and are perceived as a distraction from core work rather than being seen as a means to improve patient and client care.

Naturally there are exceptions to this and within each mental health and intellectual disability service there are users who are more technically skilled and who often act as champions for existing software and new software as it is introduced. These staff play a vital role in the take-up of new and existing software systems in St John of God Services.

2.5 Software Development in SJOG

The process of software development in St John of God Services is informal and flexible and factors such as the complexity of the project and time constraints determine the approach taken in each case. There are certain stages which are common to each software development project as will be demonstrated in the following example. This example illustrates the process from requirements analysis to product implementation for a typical software component. The Bed Management component has recently been implemented as an add on to our IDIS system. Fig 3 shows the Bed Management component called from within the main ‘Residential’ section of the IDIS system.



Residential Se
Domus
Status
Wait list
Locations
All
16 St. Fursey

Select Residential Locations
2 Domus

Select Dwelling
210 16 St. Furseys Terrac

Room No. 1
Bed 1

Room No. 2
Bed 1 Ms F LeFM
Bed 2 Ms J MoJM

🟨 Reserved bed (Internal)
🟥 Reserved bed (External)

Fig. 3 Bed Management component called from IDIS

2.51 Development of a Bed Management component.

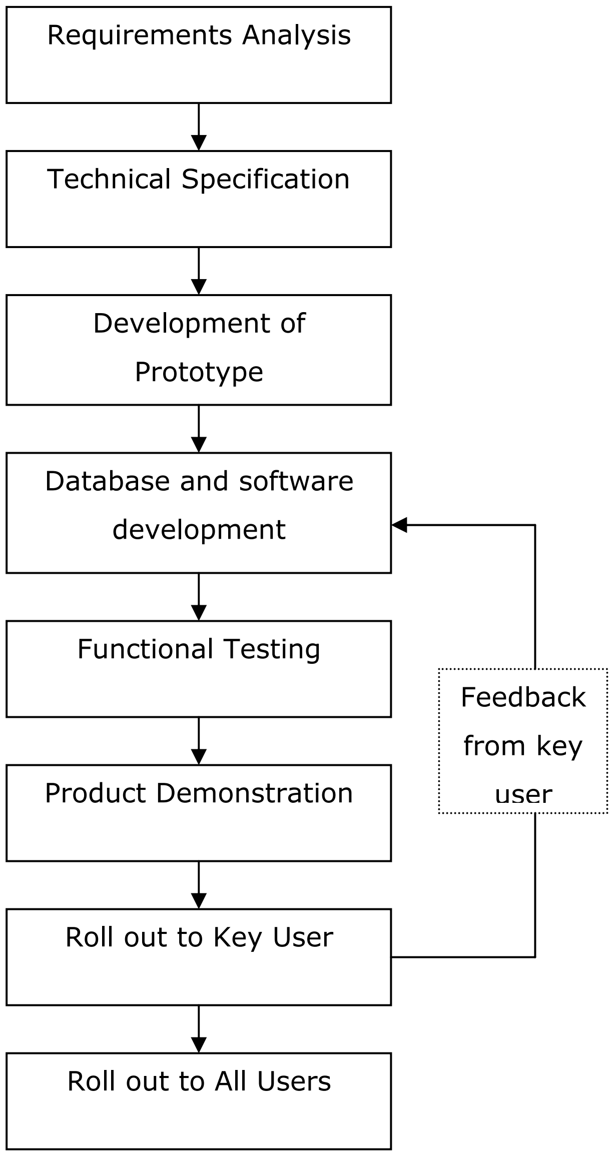
2.511 Overview

The bed management module of the IDIS system allows users to monitor bed occupancy in their service through a graphical interface. It also facilitates the moving of clients either to another bed in the service, to an external location (such as for hospital treatment) while recording a history of such moves. A similar component (wardsview) already exists in the MHIS system but key a number of key operational differences necessitated the development of a new component as opposed to an adaptation of the wardsview component.

2.512 From Requirements analysis to Implementation

Figure 4 shows each of the stages in the development and implementation of the Bed Management module and each stage is subsequently described.

Fig 4 - Software development process for Bed Management module.



Stage 1.

A meeting was held and attended by two representatives from the software development team (IT manager and one software developer) and two key users from the department requesting the component. The high level requirements were verbally presented by the prospective users and notes were taken which would form the basis of an informal technical specification document. By the end of the meeting, there was broad agreement about how the component should function but little discussion regarding usability or interface design had taken place. The meeting was concluded with a commitment from the software development team to implement the component within six weeks.

It was agreed that the component would facilitate the following transactions

A. Client A may be moved from bed 1 to bed 2.

B. Client A may be moved form bed 1 to bed 2, reserving bed 1 for client A

C. Where client A and client B are to exchange beds, provide a temporary holding to hold client B and to allow client A to move to Client B's bed. Client B can then be moved from temporary holding area to client A's old bed.

D. Where a client is temporarily transferred out of the Service (e.g. for hospital treatment), allow recording of this external move and reserve the client's bed for their return to the service.

Stage 2.

A Technical specification document was written by the software developer in consultation with the IT manager. A basic non-functioning prototype was drawn up using Visual Basic to show the various screens which would make up the finished component.

Stage 3.

A meeting was held where the software developer and one key user were present. The prototype was shown to the user and it was agreed that what was presented matched the requirements defined at the outset of the project.

Stage 4.

Following this meeting, software development and database design were commenced. As with almost all software developed in St John of God Services, Visual Basic 6 and SQL Server 7 were to be the technologies used. Through email contact, the software developer and the key user were able to resolve any queries regarding functionality which arose during development. Initial Development phase was completed with three weeks.

Stage 5.

The product was tested against a test database by the developer in conjunction with the database administrator until each of the transactions could be completed with out error being raised.

Stage 6.

By the end of the fourth week, one key user was invited to view a demonstration of the component at the software developer's workstation. Functionality of the component was demonstrated in full against a test database and it was agreed that the component would be implemented the following week.

Stage 7.

The component was implemented with access given to one key user only. Over the following week, phone calls and emails were used to communicate issues regarding functionality and usability of the component. The developer addressed each of these issues over the course of the week. The issues raised at this point are detailed in chapter 6.

Stage 8

A new version of the component was launched in week six and having confirmed that the issues raised had been resolved, it was agreed that the component access would be granted to all users.

2.513 Usability issues arising following roll out.

The component was rolled out to three users in total, one of whom was the key user. Following the implementation, a small number of issues were raised but these issues related to functionality of the component as opposed to usability.

Chapter 3: Literature Review

3.1 Overview

The purpose of this literature review is to assess the current thinking in the area of usability evaluation and to provide information which will assist in the selection of a formal usability evaluation methodology which can be applied to future software development projects in St John of God Services.

Firstly, the concepts of usability and evaluation will be discussed and the scope of the project will be established and justified. The author will then discuss the current practice regarding evaluation in health care showing the reasons why formal evaluations of software systems are rarely carried out. For a number of reasons, it is vital that the barriers which hinder the practice of software evaluation in healthcare today are overcome and these reasons will be presented in the next section.

Following this, the concepts of formative evaluation, which takes place during a product's development and summative evaluation, which is conducted following a product's implementation, will be presented. The remainder of the literature review will focus on the notion of formative evaluation as it is vital that any usability issues are addressed prior to and during implementation. Before analysing the myriad software evaluation techniques, it will be necessary to provide an overview of software development paying particular attention to the prototyping methodology which closely reflects the software development model favoured by St John of God Services.

Having discussed the working environment and how evaluation might fit with the prototyping software development model, the next step will be to consider those software evaluation models which may be applied to a healthcare setting. These techniques will be broadly categorised as descriptive, which analyse data produced following user interaction with a system and predictive techniques which employ the use of experts who aim to predict usability problems users will face when the system is implemented. Having outlined the relative merits of the many different techniques, a table will be produced which will illustrate the benefits and drawbacks of each method. The final task is to select a technique or elements of different techniques in order to produce a formal usability evaluation methodology which can be applied to future software development projects within St John of God Services.

3.2 Usability

Software can be evaluated with respect to different aspects, for example, functionality, reliability, usability, efficiency, maintainability, portability (ISO/IEC 1991) but to discuss the concept of software evaluation in relation to all of these different aspects would be beyond the scope of this dissertation so it has been decided to focus on the area of usability evaluation. The reasons for this decision are outlined below.

Usability is a vital element of software design which can mean the difference between the success and failure of a system. From the user's perspective usability is important as it can determine whether a task is completed accurately or not and whether the experience is enjoyable or frustrating. From the software developer's perspective usability is important because it can influence the credibility of a system and those who develop it. From a management point of view, software with poor usability can reduce the productivity of the workforce to a level of performance worse than without the system. In all cases, lack of usability can cost time and effort, and can greatly determine the success or failure of a system. (Usabilityfirst.com)

For these reasons, the remainder of this study will therefore deal with the evaluation of software from a usability perspective. Usability can be broadly defined as the capacity of a system to allow users to carry out their tasks safely, effectively and enjoyably (Preece & Rogers 2002). This definition is consistent with that which is contained in ISO 9241, Part 11 which defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Shakel (1991) adds user training and support defining the usability of a system as "…capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support, to fulfil the specified range of tasks, with the specified range of environmental scenarios".

Because usability is too abstract a term to study directly, it is usually divided into the following attributes (Nielsen 1993).

- Learnability
  How easy is it to learn the main system functionality and gain proficiency to complete the job.
- Efficiency
  The number of tasks per unit of time that a user can perform using the system.
- User retention over time
  It is critical for intermittent users to be able to use the system without having to climb the learning curve again.
- Error rate
  This attribute contributes negatively to usability. It does not refer to system errors but rather it addresses the number of errors the user makes while performing a task.
- Satisfaction
  This shows a user's subjective impression of the system.

A primary reason for applying usability techniques when developing a software system is to increase user efficiency and satisfaction and consequently, productivity. Usability techniques, therefore, can help users achieve their goals by helping them perform their tasks. Furthermore, good usability evaluation is gaining importance in a world where, as systems become increasingly user friendly, users are less computer literate and can't afford to spend a long time learning how a system works. Usability is critical for system acceptance – if users don't think the system will help them complete their tasks, they are less likely to accept it. It's possible they won't use the system at all or will use it inefficiently when it is implemented. If we don't properly support the user task, we are not meeting the user needs and are missing the main objective of building a software system. (Ferre et al 2001).

3.3 Evaluation

Before conducting a comprehensive review of existing literature in the area of software evaluation it is first necessary to define the concept of evaluation and to apply this definition to the world of health informatics.

Throughout the published literature there are many definitions of the concept of evaluation in relation to software engineering within the health care domain. These definitions broadly agree that evaluation comprises measuring or describing something usually to answer questions or make decisions. (Friedman and Wyatt 1990) note that most definitions of evaluation imply an empirical process where data of varying shapes and sizes are always collected. Ammenwerth et al (2004) add that simply measuring quality characteristics of an object is insufficient and that such measures need a context in which they are used: there has to be a question to be answered. They therefore define the concept of evaluation in the healthcare domain as "the act of measuring or exploring properties of a health information system (in planning, development, implementation or operation), the result of which informs a decision to be made concerning that system in a specific context."

In the past software evaluation was usually practiced following the software and development phase using statistical analysis and experimental designs but has more recently been applied using the iterative software development process. Whitefield et al (1990) note that traditional evaluations conducted in the past were "poorly integrated with development and therefore ineffective." They tended to be "too late for any substantial changes to the system to still be feasible and, in common with other human-factors contributions to development, they were often unfavourably received….this situation has been improved in recent years in a number of ways."

Any software evaluation process has predefined goals which can generally be classified in one of three ways.

- Which is better?
  The evaluation aims to compare alternative systems to choose the system which is best suited to a particular application, for a decision among

several prototypes, or for comparing several versions of a software system.

- How good is it?
  This goal aims to assess the qualities of a finished system. In this case a system may be evaluated with respect to pre-defined goals. This may be in relation to pre determined usability goals (Carroll & Rosson 1995) or it may aim to assess a system's conformity with given standards.

- Why is it bad?
  Here, the evaluation aims to determine the weakness of a software system such that the result generates suggestions for further development.

The first two goals may be considered to relate to the concept of summative evaluation whereas the third goal relates to the notion of formative evaluation.

In contrast, the goals of formative evaluation are the improvement of software and design supporting aspects (Scriven 1967). It is considered the main part of software evaluation and plays an important role in iterative development. In every development cycle, formative evaluation results in

- Quantitative data to measure the extent to which the system meets the usability goals outlined at the beginning of a software development project.

- Qualitative data which can be used to detect the usability problems of the system.

Hix and Hartson (1993) classify the resulting data by the following criteria

Objective: Directly observable data – typically user behaviour during the use of the interface or the application system

Subjective: Opinions, normally expressed by the user with respect to the usability of the interface or the application system.

Quantitative: Numerical data and results – e.g. user performance ratings

Qualitative: Non-numerical data – e.g. list of problems, suggestions for modifications to improve the interaction design.

Formative evaluation is discussed further in section 3.6.

3.4 Current practice

Before considering the potential benefits afforded to an health care organisation which engages in software evaluation, it is important to identify the barriers which can often hinder the effectiveness of such evaluations.

Rigby (2001) lists '16 powerful reasons' (see Fig 5) why the evaluation of software in healthcare is not carried out as often or as effectively as it ought to be. While this paper would appear to be more relevant to summative evaluation (evaluation conducted when the product has been implemented), many of the points can be applied to formative evaluation (evaluation conducted during the software development process). Many of the barriers identified can be broadly attributed to an innate organisational resistance to evaluation. Reasons include the diversion of resources from activities that are perceived as more creative and the reluctance to find or publicise mistakes or failures. Software Evaluation is perceived as a luxury investment where resources would be better focused on activities which directly contribute to patient care. "Evaluation is not seen as saving lives. Claims that it may increase efficiency carry significantly less weight; moreover such outcomes are only possibilities, and can only be shown retrospectively" (Rigby 2001). Below is a list of sixteen barriers as identified by Rigby.

1. No news is good news
2. Unnecessary encouragement of opposition
3. Waste of valuable time
4. Loss of credibility
5. Patient care would suffer
6. Evaluation does not save lives
7. System development would be reduced
8. It represents a professional challenge
9. Fuels inter-departmental conflict
10. Life will have moved on
11. Modern timescales are short
12. Indicators not detailed reasoning are the vogue
13. Crosses research boundaries; No "ownership"
14. Information system evaluation yields no credit
15. A liability and libel minefield
16. It is a vested interest industry

Fig 5 – 16 powerful reasons why not to evaluate. Rigby (2001)

Expanding on the notion of scarce resources, Moehr (2002) questions the quality of the sample of users at the disposal of those conducting the evaluation. Referring specifically to those methods of evaluation classified as 'Objectivist' where a sample of users is required, Moehr notes that "One has often to work with small convenience samples of highly motivated early adopters, who are not typical of the general population. Since there is often no alternative, the question is not how to achieve a representative sample but how to arrive at valid insights despite the flaws of the sample for objectivist requirements". Another inhibitor to the effectiveness of usability evaluation in healthcare, according to Moehr, is the effect of the research instrumentation on the study questions. As resources of time and people are scarce, the need to judiciously select research questions is often in conflict with time pressure. Hence, easy questions are more likely pursued than difficult ones, which however may be the important ones. Moehr (2002).

Software evaluation in health care is a complex process and is fraught with problems such as those identified by Ammenwerth et al (2004).

- Insufficiently available evaluation methods, guidelines and toolkits to cope with the complexity of health care information systems originating from a combination of technical as well as organisational and social issues.
- Insufficient collaboration between evaluation researchers from different academic fields and traditions.
- Little support by methods and guidelines for constructive (formative) evaluation in an implementation or installation project since many studies focus on summative aspects.
- Often insufficient and costly evaluation studies are carried out which do not ask or are not able to answer the important questions of information systems evaluation.

- Limited value of evaluation reports to others, because these lack sufficient information enabling others to adopt the approach or to judge the validity of the conclusions given.

3.5 Why should healthcare organisations evaluate their IT systems

Healthcare is entering the information society. Effective evaluations of health care information systems are necessary to ensure that systems adequately meet the requirements and information processing needs of users and health care organisations (Kushniruk and Patel 2004). While the use of modern information technology offers tremendous opportunities, there are also hazards associated with information technology in healthcare – modern information systems are costly, their failures may cause negative effects on patients and staff and possibly, when insufficiently designed, they may result in spending more time with the computer than with the patient (Hendrickson & Kovner 1990). Therefore, a more rigorous evaluation of IT in healthcare is recommended (Rigby 2001, Wyatt 1997) and is of great importance for decision makers and users of future information systems (Kaplan 2002).

Having outlined some of the barriers which inhibit software evaluation, one may question the value of attempting the design of, or support for, rigorous evaluation studies. Rigby (2001) explains that there are 'clear ethical and professional imperatives' which would appear to over-ride these problems, but which may yet not be making their impact, nor being heard by senior policy makers'. He sets out 6 such imperatives.

1. Organisational Duty of Care Imperative
2. Self-Review Ethical Imperative
3. Evidence-based Imperative
4. Minimisation of Disruption Imperative
5. Prevention Imperative
6. Extension and Sharing of Knowledge Imperative

Fig 6 - 6 Overriding Imperatives - (Rigby 2001)

Rigby calls his primary imperative the 'Organisational Duty of Care' Imperative. Every health care organisation has a duty of care to its patients, staff and population of potential patients. If the ultimate aim of information systems is to improve the quality of patient care then we have a duty to ensure that these systems work as efficiently and effectively as possible. If for example, the system is wasteful of time or other resources, or contains less than optimal data sets or information outputs, this can be argued as causing indirect harm, in that resources are wasted, or imperfect support is supplied to front-line staff. To monitor this situation in order to make any adjustment necessary would seem to be an essential ethical imperative.

Both the 'Self Review' Imperative and the 'Evidence Based' imperative highlight an organisation's obligation to ensure that their systems are as effective as possible. The 'Minimisation of Disruption' imperative suggests that as a system grows in terms of data and numbers of users, it becomes increasingly difficult to alter the system. The longer the system remains unchanged, the greater the disruption, retraining, data conversion and other related effects, all of which cost money, time and user patience. Structured evaluation can identify these necessary changes at the earliest opportunity.

The 'Prevention' imperative explains how many organisations draw heavily from the processes and details of those who have gone before. Structured evaluation provides a formal means of ensuring that mistakes, once identified, can be prevented from occurring in new systems or future versions of existing systems.

Finally, the 'Extension and Sharing of Knowledge' imperative notes that as technologies are moving forward with such pace and with unavoidably significant repercussions, there is a moral imperative which requires that experience and knowledge, in scientific form are generated and shared for the common good. (Rigby 2001)

Friedman and Wyatt (1997) describe five major reasons why healthcare information systems should be evaluated. The fist reason is to promote and encourage the use of healthcare information systems (HCIS) by demonstrating their safety and benefits. Secondly, evaluation of HCIS is pragmatic in that it allows developers to learn from past successes and mistakes. They also argue that there is an ethical and medicolegal obligation for evaluation. Finally they suggest that evaluation helps develop informatics as a scholarly profession by providing a foundation for research.

In the author's experience, a system's usability is a critical factor in users' acceptance of a system. Where there is often a reluctance among staff to adopt new technology, usability evaluation is an important means by which we can reduce the likelihood of users rejecting the system. This, allied to the reasons outlined above demonstrate the importance of formal usability evaluation in the development of software products in health care.

3.6 Formative and Summative Evaluation

In General, summative evaluation is concerned with the global aspects of software development, and does not offer constructive information for changing the design of a system in a direct manner. It is generally performed when the development of the system is almost or entirely accomplished. (Hix & Harston 1993)

When software evaluation was first introduced as a method for improving design, many companies would simply (summative) "test" their product just prior to shipping – a different form of quality assurance (Barnum 2002). Evaluation was conducted to determine that the product was good and provide them with the chance to claim that it is user-tested, human factor satisfied, or ergonomically sound. The problem that many organisations had was that their testing did not often reveal these results. Users would struggle with features and their implementations, and a particular design would be discovered to have numerous flaws. However, by this stage of development redesign would be too costly, and imperfect designs were delivered with improvements being stored for future implementations (Kushniruk 2002).

In recent years, therefore, an additional focus has emerged: the development of approaches to evaluation that can be used in the iterative evaluation of systems during their development (formative evaluation), with the objective of improving the design and deployment of systems as well as ensuring that the process of design of health care systems leads to effective systems (Kushniruk 2002). In the general software industry it is increasingly recognised that continued evaluation is needed throughout the system development lifecycle from early design to summative testing, in order to ensure final products meet expectations of designers, users and organisations (Kushniruk 2004, Stead 1996, Himson et al 1999, Rubin 1994). Just adding some kind of "user testing" to an existing software process is not enough – usability comes from a complete process, one that ensures usability and attests to when it has been achieved (Hix & Hartson 1993).

Given the software development method employed at St John Of God Services, it has been decided that focus should be placed on the notion of formative evaluation.

As will be demonstrated in the next section, certain software development lifecycle (SDLC) models are, by their nature, iterative and therefore lend them selves well to formative iterative evaluation.

3.7 Evaluation and the SDLC

"The key principle for maximizing usability is to employ iterative design, which progressively refines the design through evaluation from the early stages of design. The evaluation steps enable the designers and developers to incorporate user and client feedback until the system reaches an acceptable level of usability." (usabilityfirst.com)

The System Development Lifecycle (SDLC) is defined on Computerworld.com as the the overall process of developing information systems through a multistep process from investigation of initial requirements through analysis, design, implementation and maintenance.

There are many different models and methodologies, but each generally consists of a series of defined steps or stages. Analysis, design, implementation, testing and maintenance are generally part of any SDLC but traditional models such as the waterfall model (described below) are procedural in their nature with each stage commencing when the previous stage has been completed. More modern approaches entail some level of iteration which allow a design to be revisited repeatedly each time it has been evaluated.

Of the traditional models, The Waterfall Model (fig 7) is perhaps the best known.

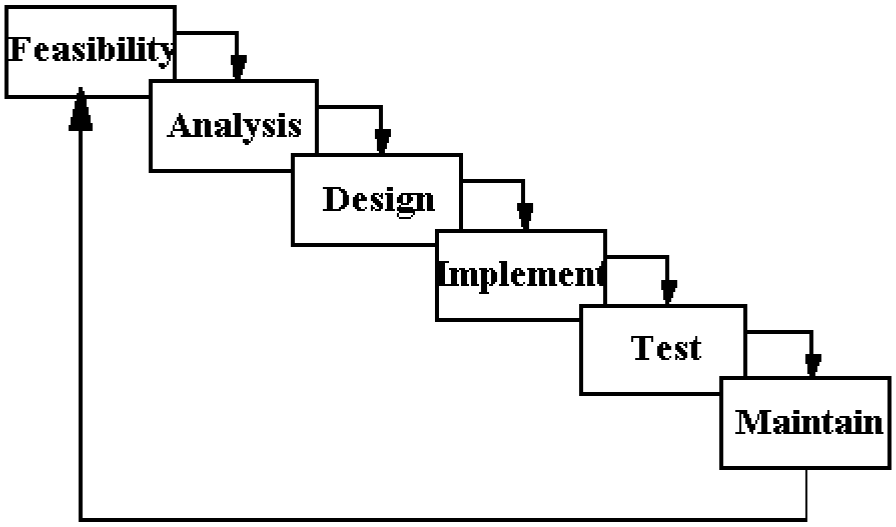


Fig. 7 Waterfall SDLC model (www.startvbdotnet.com)

The classic waterfall method implies that each subsequent step does not begin until all elements of the steps preceding it have completed, and that there is no going back to a prior step once the subsequent step has begun. (www.informit.com)

The main problem with the waterfall model from a usability evaluation perspective is that it assumes that the only role for users is in specifying requirements, and that all requirements can be specified in advance. Unfortunately, in the complex healthcare environment, requirements grow and change throughout the process and beyond, calling for considerable feedback and iterative consultation. (www.computerworld.com) . SDLC models such as the Waterfall model presuppose a set of fixed stages for system development with system evaluation being conducted primarily in the final stages. Such approaches have proven difficult to apply in health care where information needs may be hard to precisely determine. Kushniruk (2002) explains that the health care environment is often complex and characterised by missing information, shifting goals and a great degree of uncertainty. Health care decisions, he continues, are subject to a level of uncertainty not found in traditional business environments and consequently health care technology and the knowledge on which it is based are often very volatile. In fact before the decision making processes are understood, they may change within the time span of the traditional SDLC.

For many applications, design may change, particularly for highly complex and interactive applications, design may change dynamically and proceed in an iterative fashion, with feedback from end users fuelling the evolution of the design (Patel & Kushniruk 1998). The importance of design involving early and rapid prototyping has emerged, particularly regarding the design of user interfaces (Hix & Hartson 1993). Methodologies such as rapid application development (RAD) and various interpretations of prototyping have gained prominence (McConnell 1996). Usability evaluation and prototyping are interconnected. Prototypes must be developed in order to experiment with different designs and features, explore the feasibility of different aspects, but also to allow interaction with end users followed by some form of analysis. Usability evaluation facilitates the analysis of a system's prototype. When combined with usability evaluation prototyping can create an "adaptive design cycle and encourages formative iterative evaluation methods".

Design issues raised in the first evaluation are fed into the design of a further prototype and the cycle continues. (Sullivan 2004). Fig 8 shows software design as an iterative process where the design is iteratively influenced by evaluation.

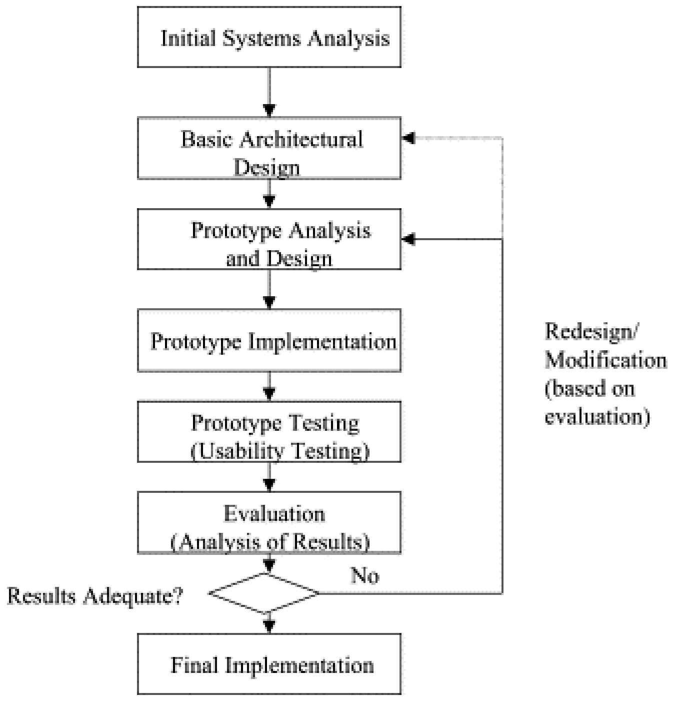

Initial Systems Analysis
Basic Architectural Design
Prototype Analysis and Design
Prototype Implementation
Prototype Testing (Usability Testing)
Evaluation (Analysis of Results)
Results Adequate?
No
Final Implementation
Redesign/ Modification (based on evaluation)

Fig 8. Iterative software design (Kushniruk 2002)

3.71 Prototyping

Prototyping is the process of designing a mock-up of some specified degree for one or more of the following purposes: to test the feasibility of an idea, clarify requirements, or allow testing/evaluation (Preece 2002). It is the chosen methodology for software development in St John of God Services.

According to Rudd (1996), there are essentially two levels of prototype, low-fidelity and high-fidelity. Low fidelity prototypes are developed quickly and at low cost and thus are suited to evaluation in the early stages of system design (formative evaluation). High fidelity prototypes are more complex and detailed and are generally used for summative evaluation. The table below illustrates the relative merits of both prototypes.

Table 3 – Relative Effectiveness of Low v High-fidelity Prototypes (Rudd 1996)

Type	**Advantages**	**Disadvantages**
Low-fidelity	Lower development costs.	Limited error checking
	Evaluate multiple design concepts.	Poor detailed specifications to code to
	Useful communication device.	Facilitator driven
	Address screen layout issues.	Limited utility after requirements established
	Useful for identifying market requirements	Navigation and flow limitations
	Proof of concept	
High-fidelity	Complete functionality	More expensive to develop
	Fully interactive	Time consuming to create
	User driven	Inefficient for proof of concept designs
	Clearly defines navigational scheme	Not effective for requirements gathering
	Use for exploration test	
	Look and feel of final product	
	Serves as living specification	
	Marketing and sales tool.	

During the iterative system development process, a number of prototypes may be utilised. The initial prototype may be as simple as a paper based sketch of a proposed user interface but as the cycle is repeated, prototypes become more complex and the final version may contain much of the functionality of the final system. The main disadvantage of the high fidelity prototype is the associated cost in terms of resources but according to Dumas and Redish (1999) this can be alleviated by using a fully functioning similar system that already exists but attacks many of the design concerns that one’s project may be facing. In evaluating such a prototype, one can reap the benefits of high fidelity prototype testing but without the associated resource cost.

Chapter 4: Methods, techniques and strategies

4.1 Overview

Many different approaches to software evaluation and in particular usability evaluation have been identified in the literature (Gediga et al 2000, Sung 1999, Bowman et al 2002). Due to the proliferation and diversity of techniques, the classification of approaches to usability evaluation has presented some difficulty in the past (Dix et al 1998, Whitefield et al 1991) and this can make any description of the various approaches appear convoluted. However, classifying these approaches is essential to achieving a comprehensive understanding of the usability evaluation techniques currently available.

Building on the ideas expressed by House (1980) who classified approaches to software evaluation as being quantitative or qualitative, Friedman and Wyatt (1997) in their seminal text in the subject area, recognize the importance of both quantitative and qualitative elements in successful software evaluation. Therefore, they choose to focus on the distinction between achieving objectivity and exploiting subjectivity in an evaluation study and refer to these different approaches as being "objectivist" or "subjectivist". (Friedman and Wyatt 1997).

The objectivist approach focuses on numerical measurement and an "attempt to obtain statistical analysis of performance or outcomes that could be considered precise, replicable and in this sense "objective" (Kushniruk and Patel 2004). The subjectivist approach is based on the concepts of personal observation and judgment, context values such as merit and worth, differing perspectives on comparison criteria, and constructive interaction among evaluators (McDaniel 2002).

Table 4: Friedman and Wyatt's classification of evaluation approaches

Classification	**Approach**	**Methodology**
Objectivist	Comparison based	Based on controlled experiments or quasi experiments, outcome indicators, planned variation.
	Objectives based	Based on a comparison between the resource to its design
	Decision facilitation	Based on questions posed by decision makers i.e. *what if* analysis
	Goal free	Based on an analysis of the resource without imposing a priori questions or goals
Subjectivist	Quasi legal	Based on a formal adversary proceeding to judge a resource
	Art criticism	Based on formal methods of criticism and judgment by connoisseurs
	Professional review	Based on a review or site visit by experts or peers
	Responsive/illuminative	Based on the observations of investigators immersed in the operating environment for a purpose of learning and understanding

Friedman and Wyatt's (1997) widely referenced text, offers a comprehensive overview of the various approaches to all aspects of software evaluation. It does not, however, pay particular attention to the notion of usability and the evaluation of same. Also, given that this study will be focusing on the notion of formative evaluation as opposed to summative evaluation, the author prefers to use the classification proposed by Gediga et al (2002) who classify evaluation techniques into two categories, the *descriptive evaluation techniques* and the *predictive evaluation techniques* both of which "should be present in every evaluation".

4.2 Descriptive techniques

Descriptive evaluation techniques are used to describe the status and the actual problems of the software in an objective, reliable and valid way. These techniques are user based and can be subdivided into several approaches:

- **Behaviour based evaluation techniques** record user behaviour while working with a system which "produces" some kind of data. Observation techniques and "thinking-aloud" protocols are examples of this approach.

- **Opinion based evaluation methods** aim to elicit the user's (subjective) opinions. Examples are interviews, surveys and questionnaires.

- **Usability Testing** is a term which was originally used in classical experimental design studies but has evolved as a technical term to imply a combination of behaviour and opinion based measures with some amount of experimental control, usually chosen by an expert.

It should be noted that all descriptive evaluation techniques require some kind of prototype and at least one user. Furthermore, data gathered by a descriptive technique needs some further interpretation by one or more experts in order to provide recommendations for future software development. Table 5 depicts the classification of the various techniques and the sections in which they will be discussed.

Table 5 : Classification of usability evaluation techniques

Descriptive			4.2
	Behaviour Based		4.21
		Observational	4.211
		Think Aloud	4.212
		Video Confrontation	4.213
	Opinion Based		4.22
		QUIS	4.221
		SUMI	4.222
		IsoMetrics	4.223
	Usability testing		4.23
Predictive			4.3
	Usability Walkthroughs		4.31
	Cognitive Walkthroughs		4.32
	Heuristic Evaluation		4.33
	Other Predictive Techniques		4.34

4.21 Behaviour based evaluation techniques

Behaviour based techniques rely on some form of observation in order to detect usability problems. Since the user is confronted with a prototype of the system, these techniques can only be applied in the later stages of system development. A common measure here is the notion of "analysis time" which compares the time taken by an expert user to complete a specific task compared to the time taken by a standard user. These techniques lend themselves well to the 'prototyping' software development model discussed above.

4.211 Observational techniques

User observation techniques aim to minimize subjectivity by using standardized procedures and documentation (Gediga et al 2002) and are conducted directly or indirectly by trained observers (Hampe-Neteler 1994). Direct observation involves observing users during task execution, with the evaluator making notes on user performance and possibly timing sequences of actions. Observational evaluation can be carried out in a laboratory or in the user's workplace (Bannon 1998). Observation techniques may yield both quantitative and qualitative data (Hix and Hartson 1993) and are often employed in a situation where the user's behaviour is of interest or if the user has difficulty articulating their behaviour while using a system (Nielsen 1993). Preece (1999) questions the merits of direct observation as the observer finds it difficult to absorb sufficient relevant information. To combat this, users' interaction with a system may be video recorded and subsequently analysed. The idea of video recording users may be as simple or as involved as desired. Indeed, Harrison (1991), in evaluating the VANNA computer system, conducted "several parallel loggings (hands, screen, face, whole body)" before beginning his data analysis. Bannon (1998) warns that observation techniques, are intrusive by nature, with the users constantly being aware that their performance is under scrutiny (be it direct or indirect) and this can alter their performance levels. Users are likely to be aware that they are being filmed. In order to reduce the impact this may have on the user's behaviour, he recommends leaving the equipment in place for several days before recording starts so the user becomes used to it. (Bannon 1998)

4.212 Think-aloud protocols

One simple yet powerful method that has been used for some time to study the usability of systems is what has been called the "think aloud" method. The method was first described within the context of usability and human computer interaction (HCI) by Lewis (1982). The think aloud method informs the evaluator of the thoughts and emotions of the user as they interact with a system (Gediga et al 2002) and has it's roots in cognitive psychology, the study of verbal protocols of subjects as they perform tasks in order to get access to the kinds of mental operations they are engaged in (Bannon 1998). The user is asked to articulate what he is thinks and what he feels while working with the prototype. The data may be recorded manually with pen and paper (Nielsen 1989) or using more modern approaches such as audio or video (Jorgensen 1989). Nielsen (1992) questions the value of technology in this instance arguing that a pen and paper approach is sufficient to elicit the relevant contextual information.

The episodes which describe the users' problems and complaints are extracted. These episodes are listed and coded by a user number and an episode number. Afterwards, the episodes are matched with one "feature" of the system. These "features" define the grouping criteria for the episodes. The design aspect of the "feature" and the frequency information provide further help for the interpretation of the results (Gediga et al 2002). By using thinking-aloud techniques, the evaluator obtains information about the whole user interface. This type of evaluation is orientated towards the investigation of the user's problem and decision behaviour while working with the system (Hempe-Neteler 1993). The procedures to transform the lingual information into evaluation criteria is only weakly standardised, which may be problematic.

Further constraints of this methods are noted by Bannon (1998). Studies are limited to a small number of people and analysis of tapes can be very time consuming. In relation to statistics, this method does not produce a lot of measures but rather gives details of complex behaviour for analysis (Bannon 1998). Gediga et al (2002) have found the expense of the method to vary depending on how the technique is implemented. Data analysis can take up to five times as long as the recording process (Gediga et al 2002).

Bannon (1998) also suggests that the notion of a person thinking aloud as they interact with a system is a bit artificial although this can be overcome by a technique known as cooperative evaluation where two people work at the interface. Communication is more natural in this scenario (Monk et al 1993)

Think aloud studies can help pinpoint problems with a system and elicit why a particular problem occurred. They can also help the evaluator understand the user’s attitude towards the system. Carroll et al (2002) found the think aloud protocol to be particularly useful in assessing the learnability of an interface with qualitative data being collected concurrently with task performance but offer a similar caveat to that expressed by Bannon that user responses may be influenced by the fact that the principle researcher is often well known to users.

4.213 Video confrontation

The video confrontation method is similar to the thinking aloud method in that users’ interactions with a system are recorded but differs in how the data is analysed. In contrast to the standard think aloud approach where a user’s involvement in the study ends when recording has been completed, with video confrontation, the user is interviewed about certain episodes which the evaluator considers interesting. The interview is guided by pragmatic or theoretical goals such as finding usability problems or examining the relationship between user errors and emotional reactions. The questions concern cognition, emotions, or problems which have occurred during the use of the system (Hamborg & Grief 1999). Questions are focused on the salient points and therefore the protocol is far easier to analyse than with the think aloud approach. Using standardized questions it is possible not only to arrive at a list of problems, but also to obtain an indication of the reasons for these problems. Other sources of information such as other recorded sequences and log-files may also help the evaluator to interpret the outcome of the procedure (Grief 1991). Bannon warns that any studies involving video recording of users are time consuming and therefore studies should be properly planned to minimize time wasted. Gediga et al (2002) also allude to the resource intensive nature of such studies where recording of information can take up to five times as long as the period which the user spent interacting with the system where time taken to evaluate the data collected can take longer again.

4.22 Opinion based evaluation methods

Opinion based methods may be either written or oral. Written evaluations take the form of structured questionnaires as opposed to oral methods which generally involve the evaluator interviewing a user or number of users to evaluate their opinion about the software under study – the interview may be structured or unstructured or, as suggested by Meister (1986) a combination of both to enable the interviewer to ask further questions as a result of the subject's statements while also conforming to a formal set of topics.

Several evaluation questionnaires are proposed in the literature (Shneiderman 1987, Yamagishi and Azuma 1987). According to Zmud & Boyton (1991), in designing any project, questionnaires should never be developed from scratch where appropriate measurement instruments already exist. The use of a standard measure with established validity and reliability allows comparison of scores with other settings and spares the evaluator the time consuming process of developing a new measure (Baroudi & Orlikowski 1988). Interviews can include scaled response questions as well as open-ended questions, thereby collecting both qualitative and quantitative data. The open ended questions may be analysed quantitatively by counting various attributes contained in the data, as in content analysis, or, as is more common in qualitative data analysis, by seeking patterns and themes. (Kaplan 1997).

The difference between oral interview techniques and questionnaire based techniques lies mainly in the effort for set up, evaluating the data, and the standardisation of the procedure. The development of an interview is more economic than for questionnaires, whereas carrying out and evaluating a questionnaire procedure can be done with less effort and costs. Standardisation and accuracy are also better for questionnaires. The advantage of an interview, such as in video confrontation, in comparison with the observational methods discussed above is that an interview helps to obtain an insight into the user's opinion of the system which cannot be gathered by observation alone (Nielsen 1993). Furthermore, these techniques are not as expensive as observational techniques. Oral interviews are held in a flexible manner after the user had come into contact with the system, and in a more structured way, if the user was faced with unforeseen aspects (Kirakowski 2000, Nielsen 1993).

Interviews and questionnaires are primarily used in the specification, design, or re-engineering phase, or as a means of system comparison (summative approach), where different techniques have different focal points. (Gediga et al). A number of standard have been proposed since Dzida et al (1978) produced their large scale questionnaire which entailed a rigorous measure of user-perceived quality. They, however, started from a consideration of system characteristics (such as input format, response time, detail of explanation etc) rather than users' expectations of and attitudes to the system being evaluated (Kirakowski 2000). Further questionnaires such as those proposed by Baily & Pearson (1983) and Lewis (1991) address the wider notion of software quality rather than focusing on usability. Two of the more prominent questionnaires currently being applied to evaluate usability are the Questionnaire for User Interface Satisfaction (QUIS) (Shneiderman 1987, Chin et al 1988) and the Software Usability Measurement Inventory (SUMI) (ucc.ie)

4.221 Questionnaire for User Interface Satisfaction

The "Questionnaire for User Interface Satisfaction" (QUIS) aims to provide a measure of overall satisfaction; additionally it evaluates some aspects of the user interface based on user opinions. The original version (QUIS 5.0, 11) consists of the following five scales each of which has attributes such as terrible/wonderful, frustrating/satisfying, dull/stimulating, inadequate power/adequate power, rigid/flexible.

- Overall User Reactions
  - Evaluates the overall reaction to the interface
- Screen
  - Evaluates the display appearance
- Terminology and System Information
  - Evaluates the effectiveness of terminology and message display
- Learning
  - Evaluates the suitability of the system for learning
- System Capabilities
  - Evaluates the efficiency and related aspects of the system in terms of speed and reliability

The current version of the QUIS includes scales relating to multimedia, online help and software installation. (Shneiderman 1998). A short (47 Items) and a long version (126 Items) of QUIS are available. Where time and resources permit, the more comprehensive version should be used but in cases where resources are scarce or when motivational problems of the user are anticipated. In the long version there are more concrete questions explaining and complementing the "leading items" (which constitute the short version) of each scale. At the beginning, there are questions about the properties of the system under study (in terms of the user's opinions) and the characteristics of the user.

The scaling of the items ranges from 1 to 9, and an additional "no answer" option. The endpoints of the scales are anchored by pairs of adjectives (e.g. difficult / easy). User comments about the system can be expressed at the end of each scale.

In the health care domain, the QUIS has been successfully applied throughout the software industry and has proven popular in the health care domain. Documented cases include the assessment of usability of physician order entry systems (Harvey et al 2001) and the evaluation of physician satisfaction with electronic medical records (Sittig et al 1999). Few studies exist which seek to evaluate the QUIS as a tool or indeed to compare it with other standardised usability questionnaires although a study by Tullis and Stetson (home.comcast.net) revealed that for small user samples, the QUIS was less effective than other methods tested. They also found little benefit in using samples of greater than twelve users. Chin et al (1988) concluded that the QUIS was generally reliable but offered no insight into the relative effectiveness of the different sub-scales.

4.222 Software Usability Measurement Inventory SUMI

SUMI is used primarily as a tool in summative evaluation and attempts to measure a user's perception of the usability of the software under study. It is the only commercially available questionnaire for the assessment of the usability of software which has been developed, validated and standardized on an international basis and it is mentioned in the ISO 9241 standard as a recognized method of testing user satisfaction. (ucc.ie).

SUMI is indicated by Preece et al (1994) as a standard method for assessing user attitudes, and by Dzida et al (1993) as a way of achieving measurement of user acceptance in the context of the Council Directive on Minimum Safety and Health Requirements for Work with Display Screen Equipment (EEC 1990)

SUMI consists of 50 items which are assigned to five scales detailed below. There is also a sixth 'global' scale consisting of 25 items which presents the perceived quality of the system as one index (Kirakowski 1993). The answer format for the items consists of "agree" and "disagree". In a SUMI evaluation, the recommended number of users is 10 to 12. The headings of the scales are:

- Global (25 items)
- Efficiency (10 items)
  Evaluates how well the software supports the user while working on the tasks.
- Affect (10 items)
  Measures the user's general emotional reaction to the software.
- Helpfulness (10 items)
  Measures the degree to which the software is self-explanatory, and also the suitability of the help system.
- Control (10 items)
  Measures the degree of the user's feeling the (s)he controls the software.
- Learnability (10 items)
  Measures time and effort for learning the handling of the software from the user's point of view.

Originally applied to summative evaluation studies, SUMI was supplemented by the "Item Consensual Analysis" (ICA) so that it could be used as a tool of formative evaluation. The ICA enables the evaluator to locate usability problems more precisely than with the analysis of the scales profiles. ICA requires a "Standardisation Database" consisting of expected pattern of responses for any SUMI item. A comparison of expected and observed frequencies for the items (using a Chi_test) shows which item signals a demand for change.

The minimum user sample size needed for an analysis with tolerable precision using SUMI is in the order of 10-12 users although evaluations have been carried out successfully with smaller groups.

The effectiveness of the questionnaire is more likely to be affected by the care with which the context of use of the software has been studied and whether or not a design plan has been drawn up. Validation studies of SUMI have shown that the questionnaire has the capability to distinguish software of different ergonomic quality. The usefulness of SUMI in consultancy-based studies as well as in case studies has been exemplified in (Kirakowski 2000). Whereas earlier questionnaires were usually applied as tools in summative testing, Kirakowski points out that the SUMI questionnaire lends itself ideally to iterative development models as it is short (five minutes to complete) and does not require a large sample (Kirakowski 2000, Redmond-Pyle & Moore 1995).

4.223 IsoMetrics

The IsoMetrics usability inventory provides a user-oriented, summative as well as formative approach to software evaluation and is based on ISO 9241 (Part 10). ISO 9241 deals with the "Ergonomics of Human System Interaction". Part 10 which focuses on dialogue principles presents a set of usability heuristics that applies to the interaction of people and information systems. The standard refers to this interaction as a "dialogue" and describes seven "dialogue principles"

- Suitability for the task (17 items)
  The dialog should be suitable for the user's task and skill level.
- Self descriptiveness (14 items)
  The dialogue should make it clear what the user should do next
- Controllability (14 items)
  The user should be able to control the pace and sequence of the interaction
- Conformity with User Expectations (9 items)
  It should be consistent
- Error Tolerance (17 items)
  The dialogue should be forgiving
- Suitability for Individualisation (11 items)
  The dialogue should be able to be customized to suit the user
- Suitability for learning (8 items)
  The dialogue should support Learning

The inventory consists of 151 items which are assigned to the various principles. The IsoMetrics usability inventory is available in two versions both of which are based on the same pool of items but using different formats to allow summative as well as formative evaluation procedures. IsoMetrics^S (short) is a summative evaluation tool whereas IsoMetrics^L (long) is suited to formative evaluation. IsoMetrics^L contains a five point rating for each of the items starting from one (predominantly disagree) to five (predominantly disagree). A further category (no opinion) is offered to reduce arbitrary answers by the user. Unlike the short version, users are asked to give a second rating where they rate the importance of the item in terms of supporting their general impression of the software. (Gediga et al 1999).

This rating ranges from 1 ("unimportant") to 5 ("important"), and a further "no opinion" category may also be selected. In this way, each item is supplied with a weighting index. The data collected may then be analysed using a range of statistical analysis techniques.

The IsoMetrics design provides information that can be used within an iterative software development. There are a number of features of the inventory which make it suitable as an evaluative tool in an iterative software development environment. These include scores of the usability dimension to measure the progress of development, concrete information about malfunctions and their user-perceived attributes and measures such as mean weight of any user-perceived attribute, given a class of system malfunctions.

Isometrics^L helps identify weak points of software systems, and therefore provides concrete impact on their improvement and redesign. Since the inventory evaluates software from a user's perspective, it supports a participative and user-oriented approach of system design. (Gediga et al 2000).

IsoMetrics has proven effective in a number of software development projects. Gediga et al (1999 b) conducted two reliability studies for five software systems and found that the IsoMetrics inventory could be justified for each of the seven design principles. Hamborg et al (2004) showed that Isometrics is a well suited and reliable technique in the area of health information systems. They note, however, that the Isometrics questionnaire provides hints as to problem areas as opposed to concrete weaknesses and suggest combining it's use with other evaluation methods for best results.

4.23 Usability Testing

Usability testing is a technique for gathering empirical data by observing users as they perform tasks with the application that is being evaluated. Usability testing may be conducted in the field but is more commonly conducted in a usability laboratory where equipment for recording and observing the sessions is available. (Consolvo et al 2003). Usability Testing uses a systematic and quite rigid experimentally based gathering of information about a product or a prototype using user representatives (Rubin 1994). There are numerous approaches to usability testing discussed in the literature. Rubin (1994) loosely categorises the different approaches:

Firstly, formal tests conducted as true experiments. This approach is characterised by the use of classical experimental designs for testing hypotheses and for deriving causal dependencies. The second, less formal class of usability tests employ an iterative cycle of tests intended to expose usability deficiencies, and gradually shape or mould the product in question.

The latter approach differs significantly from classical experimental designs in terms of the accuracy of the description of the independent factors. However, the problem of defining the variables to be measured exists for both approaches.

Kushniruk and Patel (2003) outline the nine phases involved in applying usability testing approaches to the evaluation of software systems.

Phase 1 – Identification of evaluation objectives
Phase 2 – Sample selection and study design
Phase 3 – Selection of representative experimental tasks and contexts
Phase 4 – Selection of background questionnaires
Phase 5 – Selection of the evaluation environment
Phase 6 – Data collection – video recording and recording of thought processes
Phase 7 – Analysis of the process data
Phase 8 – Interpretation of findings
Phase 9 – Iterative input into design

The dependent variables are chosen pragmatically according to the evaluation goals. Techniques described above, including questionnaires and interviews, observational methods, think aloud technique and video-confrontation are used to measure the impact of differences in system design, or different versions of a prototype on the usability of the product.

Usability testing also employs the use of measurement criteria such as "time to complete a task" or "percentage of tasks completed", which can be easily applied, if tasks are accurately described. (Tyldesley 1988).

The think aloud technique appears to be the technique most commonly used for qualitative data generation. (Hix and Hartson 1993). Nielsen (1993) agrees that "User testing with real users is the most fundamental usability method and is in some sense irreplaceable, since it provides direct information about how people use computers and what their exact problems are with the concrete interface being tested".

While this method can prove very effective, it is quite costly to conduct when compared with other methods. Wharton et al (1991) concluded that usability testing was an effective means of identifying serious and recurring problems, and avoided identifying low-priority problems, but was the most expensive testing method. Inn spite of the expense involved in conducting such studies, Karat (1993) found that when compared to heuristic (expert) evaluations, the cost on a per-problem basis was actually lower with usability testing.

Virzi et al (1993) concluded that the think-aloud approach seems to be desirable for products that can be tested with readily available subject populations. If obtaining naïve subjects is expensive e.g. where the desired population is geographically dispersed, the costs of this type of evaluation will increase making other approaches more attractive in comparison.

Although it has been argued that a significant disadvantage of usability testing is that the testing environment tends to be artificial and that users may behave differently knowing they are being watched (Consolvo et al 2003), it is generally agreed that when compared to studies which employ the use of evaluation by experts (heuristics), a usability test identifies the problems that will plague the actual users of the application. Developers may doubt that a problem exists, but when they see the user actually experience the problem in the laboratory, they change their minds quickly. (Jeffries and Desurvire 1992).

Because Usability Testing requires a large amount of expertise to set up the experimental design, choose the suitable tasks for comparison, select the users and the number of users, define the measurables properly, it is perhaps best suited for usability engineers. It is certainly not a suitable technique for untrained evaluators. (Gediga et al 2000)

4.3 Predictive techniques

Unlike observational and opinion based evaluation methods which require a more or less sophisticated prototype, predictive evaluation techniques do not require a built system. Empirical methods as employed in the descriptive techniques outlined above are replaced by a theory on the contribution of experts. Consequently, user involvement is not as dominant as in the empirical evaluation techniques. Despite this, user participation can be more prominent in predictive approaches as user representatives have the opportunity to actively influence the software development process whereas with the descriptive techniques, the user plays a more passive role. (Gediga et al 2000)

4.31 Usability Walkthroughs

Like the usability testing approaches, usability walkthroughs are based on the concept of task analysis, where the evaluation is conducted in the context of particular information processing tasks which are defined at the outset. However, inspections are not based on empirical testing of end users of a system, but rather a trained analyst (or team of analysts) steps through and simulates the task under study. This approach entails the methodical analysis of an interface where problems or cognitive issues are noted as the analyst(s) "walks through" the system in order to carry out the particular task. (Kushniruk and Patel 2004)

As well as a trained analyst, walkthroughs may involve a group of participants such as representatives from different disciplines, expected users, product developers and human factors specialists. Group based usability walkthroughs have proven to be more effective than individual walkthroughs. (Preece 1999)

Walkthroughs are particularly useful for identifying problems caused by the gap between system behaviour and user expectation (Wharton et al 1994) and are also effective in determining whether the user interface is perceived to be adequate focusing on such details as wording, the distinction between buttons , commands or menus and the analysis of preset tasks. (Gediga et al 2000)

Due to the associated cost, usability walkthroughs tend not to focus on the entire interface but rather on selected features. Other disadvantages of this method noted in the literature include the fact that the success of the walkthrough is a function of the combination and the psychological fit of the group members (Karat 1997). Furthermore, explorative behaviour can only be simulated in a limited manner so unexpected errors and glitches cannot be detected by usability walkthroughs. (Bias 1994). Probably the most popular approach to usability evaluations of this nature is the Cognitive Walkthrough.

4.32 Cognitive Walkthrough

The cognitive walkthrough approach entails the identification of sequences of actions and subgoals for successfully completing a task and assigning causes to usability problems. The approach focuses on evaluating how well a task can be completed while using a system and can therefore be considered a form of task analysis. (Kushniruk and Patel 2004). The method seeks to ascertain how easy a system is for a user to learn and involves evaluating the system in the context of specific user tasks. The cognitive walkthrough involves answering a set of questions about each of the decisions the user must make as they use the interface. The questions are concerned with identifying the users' goals, the ease with which users will be able to identify the consequences of a decision and how easy it is for users to evaluate whether they are progressing towards a goal. These questions are asked for each step of a task and following each task, the likelihood of users having problems making the correct choice is rated. (Kushniruk and Patel 2004).

According to Bannon (1998), one of the advantages of this method is that it makes the users' goals and expectations explicit. Another advantage of this method is that it can be applied in early stages of the development, and that not only usability weaknesses of the system are reported, but other problems as well. (Gediga et al 2000).

The Cognitive Walkthrough method requires an accurate and thorough understanding of who the users are and what skills and experience they have (Bannon 1998). Gediga et al (2000) add that the analyst should be familiar with at least the basics of system development. It should also be noted that it is a tedious method to perform and is less effective than other methods such as heuristic evaluation. (Bannon 1998). The CW method has also been criticised in that the focus of analysis sacrifices other important usability information such as overall consistency (Vizri 1997).

4.33 Heuristic evaluation

Heuristic evaluation is a usability inspection method in which the system is evaluated on the basis of well established design principles such as visibility of system status, user control and freedom, consistency and standards, flexibility and efficiency of use (Kushniruk and Patel 2004). The 10 heuristics developed by Jacob Nielsen (1993) were applied to health informatics by Zhang et al (2003)

1. **Visibility of system status**
   The user should be kept informed of system status at all times while they interact with the system. E.g. When a task is in process or when it has been completed, appropriate feedback should be given to the user.
2. **Match the system to the real world**
   The system should use natural language at all times to communicate with the user and should use 'real life' metaphors to map tasks to real world conventions. For example, having a 'rewind' button to indicate backward navigation maps to the physical rewind button on a common VCR or cassette recorder.
3. **User control and freedom**
   The user should always feel that they are in control of the program. Offering clearly marked exits, supporting undo and redo transactions and making it difficult to perform irreversible actions.
4. **Consistency and standards**
   The user interface and basic system functions should be consistently used throughout the system.

5. **Error Prevention**
   The interface should be as simple and straight forward as possible so as to minimize the likelihood that the user will encounter an error in using the system.
6. **Minimise memory load – support recognition rather than recall.**
   In order to process a task, the user should not be required to remember a series of steps, rather the system should make it easy for the user to recognize the steps required.
7. **Flexibility and efficiency of use**
   Users should be able to tailor the system so that it suits their individual needs. Examples include allowing the use of shortcuts and the colour coding of different elements of the interface.
8. **Aesthetic and minimalist design**
   User interfaces should be kept as simple as possible and the use of superfluous features should be avoided.
9. **Help users recognize, diagnose and recover from errors**
   Where an error has been encountered, the user should be provided with clear and easy to understand information about how to recover from the error encountered.
10. **Help and documentation**
    Help should be available throughout the system to assist the user with the completion of tasks. Naturally, this help should be easy to navigate and to follow.

The method employs multiple evaluators who conduct independent inspections, comparing user interface elements against the usability heuristics outlined above. The results from the experts involved are combined in order to maximise the chances of properly identifying any usability problems and are then ranked to prioritise iterative (re)design of each usability issue identified. No representative users are included in the evaluation process (Bowman & Gabbard 2002). Studies have found that the use of three to five evaluators is the reasonable minimum that will ensure identification of about 75% of usability problems in a project. The involvement of further evaluators has been found to only marginally improve the problem detection rate (Nielsen 1994). According to Jeffries et al (1991), the availability of skilled user interface specialists is a significant hindrance to the successful employment of this method in many organisations.

Nielsen (1995) defending heuristic evaluation in this regard stresses the flexibility of the method finding that it exhibits "graceful degradation", implying that small deviations from recommended practice only results in slightly reduced benefits. Indeed, he cites a number of instances where heuristic evaluation has proven valuable where only one evaluator is involved. A further advantage according to Nielsen is that it can be applied at all stages of product development from paper mock up to advanced prototype.

4.34 Other Predictive Techniques

As well as the techniques outlined above there are other predictive techniques which deserve mention. While the walkthrough methods described above have been shown to be effective as a supplement to empirical testing methods (Lewis et al 1990) structured walkthrough procedures tend to be time consuming and unpopular with evaluators when used on substantial tasks. (Rowley and Rhoades 1992) In an attempt to maximise the problem detection rate while minimising the overhead associated with the procedure, a number or new techniques have been proposed.

The cognitive jogthrough is an example of a fast paced methodology developed for use within the constraints of a real world product development environment. (Rowley and Rhoades 1992). Rowley and Rhoades (1992) found that, when compared with the results of the cognitive walkthrough, the feedback obtained in the jogthrough was broadly similar but far more actions were evaluated and design suggestions made that were not well accommodated by the rigid structure of the walkthrough procedure.

The graphical jogthrough is a further modification of a standard jogthrough method where the ratings of evaluators produce evidence in the form of a graph, displaying estimated proportion of users who effectively use the interface versus the time they had to work with it in order to achieve the effectiveness. Demetriadis et al (1999) having applied their method to the evaluation of an interface for a network simulator conclude that the graphical jogthrough can offer useful quantitative and qualitative feedback to designers because the graph encourages evaluators to focus on the process of the user gradually gaining familiarity with the interface tasks.

Software evaluation using the ISO 9241 Evaluator was developed in response to an increasing need for practical and comprehensive evaluation methods and tools for conformance testing with ISO standards. As with the ISOmetrics observational method described above, this technique refers to the ISO Standard 9241 - the ergonomics of human-system interaction. The ISO 9241-Evaluator is a guideline orientated expert-based evaluation method that prepares the requirements of the standard to be tested in approximately 300 test items. The items are structured in a two dimensional space defined by technical components and software-ergonomic criteria where the individual technical components are rated in regard to the principles of ISO 9241. (Oppermann et al 1997)

4.4 Comparison of techniques

"In the case of usability, doing something is almost always better than doing nothing. However, for HCI practitioners, making choices based on misleading or erroneous claims can be detrimental – compromising the quality and integrity of the evaluation, incurring unnecessary costs, or undermining the practitioner's credibility within the design team." (Gray and Salzman 1998).

Gray and Salzman (1998), reviewing five studies which attempt to compare evaluation methods, outline two major failings in such studies. Firstly, they contend that those papers reviewed have adopted methods and statistical tests that are inadequate to demonstrate cause and effect. This may be due to the inherent difficulty in conducting well controlled research in an applied setting although they cite the work of Vizri et al (1993) among others as being an exception to this and proof that it is not impossible. Their second criticism centres around the notion of "generality, primarily the construct validity of effect". Outcomes of interest may be very different for different types of system – e.g. for safety critical systems, the outcomes of interest are very different from those of ATMs or video games.

Despite the concerns expressed by Gray and Salzman (1998), it is important to determine which usability evaluation methods are best suited to the system being evaluated and the environment in which the evaluation is being conducted. A number of studies have been conducted which compare heuristic evaluation against other predictive evaluation methodologies (John & Marks 1996, Nielsen 1995), and against observational methods (Jeffries & Desurvire 1992, Virzi et al 1993, Wharton et al 1991, Simeral and Russell 1997) and walkthrough techniques versus observational methods (Karat et al 1992, Rogers et al 2005).

Desurvire et al (1992) identified a number of usability problems with a product in a lab through usability testing and then compared the techniques of heuristic evaluation and cognitive walkthrough to see which technique found most of the problems identified. Heuristic evaluation, when conducted by experts, was shown to be more effective than cognitive walkthrough conducted by experts at identifying problems. However, when the same comparison was conducted by non-experts, the heuristic evaluation method was equally effective as the CW at spotting the problems found in the lab.

Comparing heuristic evaluation with usability testing, Simeral & Branaghan (1997) argue that usability testing approximates the user's initial experience of using a product without assistance, employing real users conducting real tasks in an environment similar to their home or office. Also, through this method, once a problem has been identified, the analyst is given a good idea of the severity of a problem from a user's perspective - usability problems which prevent users from proceeding are naturally more severe than those which simple annoy the user. This is consistent with the findings of Jeffries and Desuvire (1992) who found, in all of their studies, that "problems identified in the usability test were above the median in severity". This view was echoed by Karat et al (1992) who found that empirical methods were significantly more effective than walkthrough methods at identifying serious usability problems. In relation to the type of problems found, Jeffries et al (1991) note that usability testing was highly effective at finding serious usability problems when compared to a number of predictive techniques.

Another significant benefit of usability testing over predictive techniques according to Jeffries and Desurvire (1992) is that problems identified by users "have an impact on engineers developing the product that no expert evaluation can equal." They may doubt that a user interface problem exists but when they see users encounter the problem in a laboratory situation, any doubts are shed. This is echoed by John and Marks (1996) who, in comparing the relative impact of six different predictive evaluation techniques on software development, found that up to thirty percent of usability problems reported to software developers, following a usability evaluation, were not perceived by the developers to warrant design modification.

It is clear from the limited literature which compares different techniques that there are clear advantages associated with each of the many different approaches. Thus, in order to draw on the all the benefits offered by the descriptive and predictive techniques, it would seem that a successful usability evaluation strategy will ideally include techniques from both. According to Jeffries et al (1991), deciding among the various techniques on offer "requires careful consideration of the goals of the evaluation, the kinds of insight sought and the resources available."

Table 6 – Summary of usability evaluation techniques

Technique	**Advantages**	**Disadvantages**
Behaviour based *record user behaviour while they use a system*	- record data based on real users interacting with a system - meaningful problems identified	- Expensive to implement - Time consuming to analyse data - Require a prototype
Opinion based techniques *user opinion gathered through interview / questionnaire*	- Inexpensive to distribute - Structured data easier to analyse - Can identify user perceptions - Questionnaires commercially available	- Rigid structure may miss important information. - Response rate often low.
Usability testing *experimentally based gathering of data from user representatives*	- measurement criteria easier to analyse. E.g. time taken to complete task. - suitable for comparing alternative interfaces	- Expensive to conduct - low cost benefit relation - Expertise required
Usability walkthroughs *to evaluate a product from the perspective of the end user*	- Easy to learn and use - Facilitates iterative testing	- Costly to conduct - Only selected features considered
Cognitive walkthrough *assesses the usability of the user interface*	- Makes the users goals and expectations explicit	- Requires accurate and thorough understanding of user skill and experience
Heuristic Evaluation *use a predefined list of heuristics to find usability problems*	- Easy to learn and use - Inexpensive to implement - Can identify problems early in design process	- Debriefing session required to find the indication of how to fix problems - Can't replicate real user behaviour

Chapter 5: Choosing an Evaluation Technique.

5.1 Set the Goals for the Evaluation

Before selecting an evaluation technique for a particular project, it is important to clearly identify the goal or goals of the evaluation. This is the most important aspect of the evaluation and failure to set goals clearly at the outset may result in incomplete or invalid information being gathered.

Firstly, we must decide what is being evaluated. Is the purpose of the evaluation to evaluate performance, functionality, usability and or other aspects of the system or is it to focus on one area? The purpose of this dissertation is to propose a usability evaluation strategy for software development in St John of God Services so the primary goal in our evaluation strategy will be to evaluate the usability of each software system or component being developed.

Referring back to the notion of usability, as discussed in the literature review, we will evaluate each system's

- Learnability:
  How easy is it to learn the main system functionality and gain proficiency to complete the job.
- Efficiency:
  The number of tasks per unit of time that a user can perform using the system.
- User retention over time:
  How easy it is for the user to become familiar with the workings of the system
- Error rate:
  The number of errors the user makes while completing a given task.
- Satisfaction:
  The user's subjective impression of the system.

Having outlined the goals we hope to achieve through the implementation of a usability evaluation strategy at St John of God Services, it is next necessary to identify the factors which might influence the selection of a usability evaluation technique and to assess how significant each of these factors is to the St John of God case.

5.2 Select an Evaluation Technique

In order to achieve these goals, it is necessary to choose a usability evaluation technique which best suits the specific requirements of the software development department in St John of God Services given the resource constraints outlined in chapter 2.

In reviewing the literature, the author was unable to source a definitive list of factors to consider in choosing a usability evaluation technique. However, drawing from the literature review, it is felt that the following list represents those factors which an organisation might consider in selecting the usability evaluation technique most appropriate for their specific needs. Below each point is a statement of how relevant each is to SJOG Services.

- **Cost of Documentation**

  Certain evaluation techniques require the purchase of documentation on which the evaluation is based. For example, any evaluation which is based on the ISO 9241 (Ergonomics of Human System Interaction) is likely to involve a significant outlay in order to purchase the complete standard from the ISO website.

  Relevance to SJOG

  Funding for new IT related endeavours is limited and whatever technique was selected would have to be a low cost option. One option considered was to purchase the complete standard, ISO 9241, and tailor a bespoke predictive evaluation technique based on it. The cost of purchasing the standard precluded this option.

- **Availability of Equipment**

  The observational techniques described in chapter 3 advocate the video recording of users as they use a system. Choosing such a technique requires access to such equipment as well as the availability of a physical environment in which to conduct the session.

  Relevance to SJOG

  Any usability evaluation technique which entails the use of audio visual equipment and preferably a lab in which to make the recordings, can not be considered. Neither the equipment, the skills to operate it nor the expertise *to analyse what is recorded are presently available to the software development department in SJOG.*

- **Level of Expertise Required / Ease of Learning**

This is a factor which differs greatly among the different techniques described in the literature review. For example, a number of the techniques discussed require the participation of usability experts who are skilled both in administering the technique as well as in analysing the data produced. In contrast, some techniques can be easily learned and carried out effectively by staff who are not necessarily skilled in the field of usability. The heuristic review technique described in chapter 3 is an example of such a technique.

Relevance to SJOG
Of the software development team members described in chapter 2, none could be described as having expertise in the area of usability. Techniques which require expertise in this regard would have to be excluded form the selection process. Also, certain techniques such as the IsometricsL technique, require skills in the area of statistical analysis which are not present in the IT department. Ease of learning is vital, not only because of time pressures but also, new staff members must be quickly able to familiarise themselves with the technique. Nielsen's heuristic review is particularly attractive in this regard.

- **User Participation**

While all of the descriptive techniques described in the literature review require the involvement of users, there are many usability evaluation techniques available which do not. While the involvement of users in any usability evaluation strategy is always desirable, it is not always feasible. Geographical location of users, their availability to participate as well as their willingness to participate are just some of the factors which need to be considered before deciding whether to opt for a technique which can be conducted without user involvement or one which relies on user participation.

Relevance to SJOG
As was explained in chapter 2, the users of computer systems in SJOG are spread throughout Ireland and this represents a significant obstacle to any technique involving users. Coupled with this, finding users who are available and willing to participate is difficult in an environment where computer

systems are often viewed as a distraction from core work. Naturally there are exceptions to this and for each system or component developed there is generally at least one key user who is happy to assist in any way in order to improve the quality of the final product. As was mentioned in the literature review, it is widely accepted that the most effective evaluation strategies exploit the merits of both the predictive and descriptive approaches and so despite the difficulties expressed, it is the author's view that to completely exclude users from any usability evaluation strategy is not an acceptable route.

- **Effectiveness in identifying usability problems**

As detailed in chapter 3, certain usability evaluation techniques have been shown to be more effective at identifying usability problems than others. Different techniques identify different types of problems, some can identify large numbers of insignificant problems while others identify small numbers of important issues and different techniques produce different results depending on how they are conducted.

Relevance to SJOG
With the main systems (MHIS and IDIS) already operational, most software development work being carried out now and in the future will involve the development of new components such as the Bed Management component described in chapter 2. Such components are generally quite small in terms of number of screens and extent of functionality. Therefore it is felt that positive results can be obtained by using a technique which my be deemed inadequate in a different setting. While a comprehensive multi-technique strategy would doubtless identify more issues, such as approach would be impracticable given the resource constraints outlined in chapter 2. Techniques such as Nielsen's heuristic review, while limited somewhat by their inherent simplicity, would be very effective if executed properly in the SJOG environment given the nature of the software being developed.

- **Localisation of Evaluation Technique**

In large international organisations, it is important to know whether the technique being considered can be easily adapted to different environments. For example, if an international organisation chose to implement the Software Usability Measurement Inventory (SUMI) described in chapter 3, it would be important to establish whether the inventory was available in different languages and whether the items shared the same relevance across different geographical locations.

Relevance to SJOG
As the software development team is based in Dublin and developers are English speaking, this factor is not significant for the purposes of choosing an evaluation technique for SJOG Services.

Considering the above factors, it is the authors conclusion that the most appropriate usability evaluation strategy for SJOG Services to pursue would incorporate both a predictive and descriptive element.

The predictive technique the author has opted for is Nielsen's Heuristic Review. The following are the main reasons why this choice has been made:

- There is no cost associated with purchasing the evaluation instrument.
- The technique is easy to learn.
- It does not require the participation of users.
- It is relatively simple to conduct the review and to analyse the resultant data
- It has been shown to be effective when deployed by a small number of evaluators who do not require any specific knowledge in the area of usability.

In order to supplement this technique, it has been concluded that some level of user based evaluation is vital. As was described in chapter 2, informal user based evaluation is already commonplace in software development in SJOG Services. The level of user involvement in a project is generally influenced by factors such as the complexity of the project as well as the user related issues outlined above. Unlike the application of Nielsen's Heuristic Review, any attempt to formalise user based evaluation is difficult as the extent of user involvement varies greatly between projects as does the number of design iterations.

With paper or throwaway prototypes which are produced early in the design cycle, feedback is informally recorded through notes taken meetings attended by IT staff and key users. However, when a project reaches a stage where the basic elements of functionality are in place, a session will take place with a key user and the software developer whereby the user can step through a number of key tasks while providing feedback to the developer who will note the usability issues and following the session will apply the notes to the heuristics structure.

Chapter 6: Heuristic Evaluation Trial

6.1 Overview

Heuristic evaluation is a usability engineering method used to identify usability problems in the design of a user interface so that they may be solved by developers as part of an iterative software design process. (www.useit.com).

As was mentioned in chapter 2, one key user was on hand to raise usability issues during the design cycle. There was, effectively, one iteration in the design cycle where user feedback was integrated into the design. The starting point for this trial was the same point at which the feedback of the key user was first sought. This enabled a comparison between the types of issues raised by the key user against those raised by the evaluator.

As with all the predictive techniques discussed in chapter 4, the involvement of more than one evaluator is advocated. In the case of the software development in St John of God Services, this is not feasible due to the unavailability of resources. Nielsen notes that through careful and thorough application of the technique, significant results can be achieved with the involvement of just one evaluator. (www.useit.com).

As mentioned in chapter 2, there are two full time software developers working in St John of God Services. Developers usually work on separate projects and if and when there is a combined effort in the development of a software component, there is no collusion in the design of individual screens. Both developers have read and become familiar with each of the ten heuristics. It is proposed that the developer of the user interface (developer) will, following a briefing session, hand over the component to the second developer (evaluator) who will carry out the evaluation. The evaluator will see the user interface for the first time at the briefing session. In the briefing session, the developer will talk the evaluator through the functionality and flow of the component to be evaluated. Once the evaluator has gained sufficient understanding of the functionality of the component, they are presented with a task list which is composed by the developer to ensure that each dialogue within the system is evaluated.

The evaluation is then commenced and the tasks on the task list are completed by the evaluator. The output from using the heuristic evaluation method is a list of usability problems in the interface with references to those usability principles that were violated by the design in each case in the opinion of the evaluator. (www.useit.com). The ten usability heuristics developed by Nielsen (www.useit.com), described in section 4.33 are listed below.

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

All but the last of the heuristics are relevant in this sample case. An online manual has been developed as a separate project and thus the 'help and documentation' heuristic was not considered as part of this evaluation.

6.2 Usability evaluation of Bed Management component

The Bed Management component as described in chapter 2 is launched from within the main Intellectual Disability Information System (IDIS). A list of tasks was prepared and the list was presented to the evaluator following the briefing session. Instruction was given as to how the component should be launched. This was not part of the task list. The screen shots were not offered as part of the task list but are included here for illustrative purposes. The developer was on hand to answer any questions regarding the completion of each task. Tasks 1-5 were completed from the main screen and tasks 6-8 were completed from the holding area screen. (see fig 9 for list of tasks)

The version of the Bed Management component which was used for the purposes of the evaluation was the same version of the component which was reviewed by the key user during the product's development. As the component was quite incomplex, a single iteration in the design cycle is all that was required. More complex systems and components may require multiple iterations as shown in section 2.512.

While the component had been implemented prior to the heuristic evaluation being conducted, many of the issues raised in the evaluation were resolved by the developer and a further release was implemented.

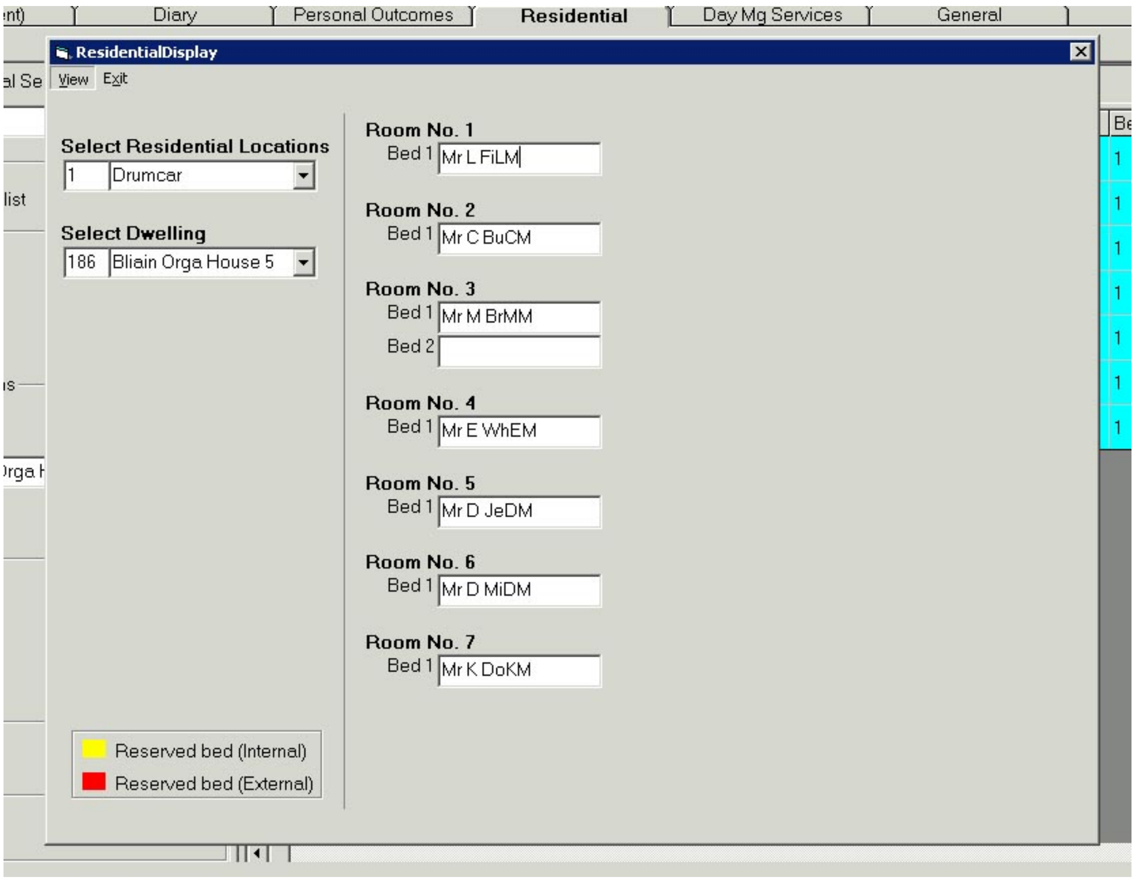
Section 6.21 details the steps involved in launching the Bed Management component from within the IDIS system and is followed by a list of tasks which the evaluator was asked to perform so that all screen elements could be evaluated. The 8 steps are listed below.

1. Vacate bed and move client direct to an empty bed
2. Reserve bed and move client direct to an empty bed
3. Vacate bed and move client to holding area (internal transfer)
4. Reserve bed and move client to holding area (internal transfer)
5. Reserve bed and move client to holding area (external move)
6. Move client from holding area to empty bed
7. Return client to reserved bed (internal)
8. Return client to reserved bed (external)

Fig 9 – list of tasks to be compled by evaluator.

6.21 To launch Bed Management component

- Launch IDIS system
- Select 'Residential' tab
- Select a 'Managed Service'
- Set 'Status' = 'Current'
- Set 'Location' = 'Specific'
- Choose a Dwelling – with > 5 clients in residence and >= 1 empty bed
- Click 'Dwelling Graphic' on top of IDIS main Screen



Select Residential Locations: 1 Drumcar

Select Dwelling: 186 Bliain Orga House 5

Room No. 1
Bed 1 Mr L FiLM

Room No. 2
Bed 1 Mr C BuCM

Room No. 3
Bed 1 Mr M BrMM
Bed 2

Room No. 4
Bed 1 Mr E WhEM

Room No. 5
Bed 1 Mr D JeDM

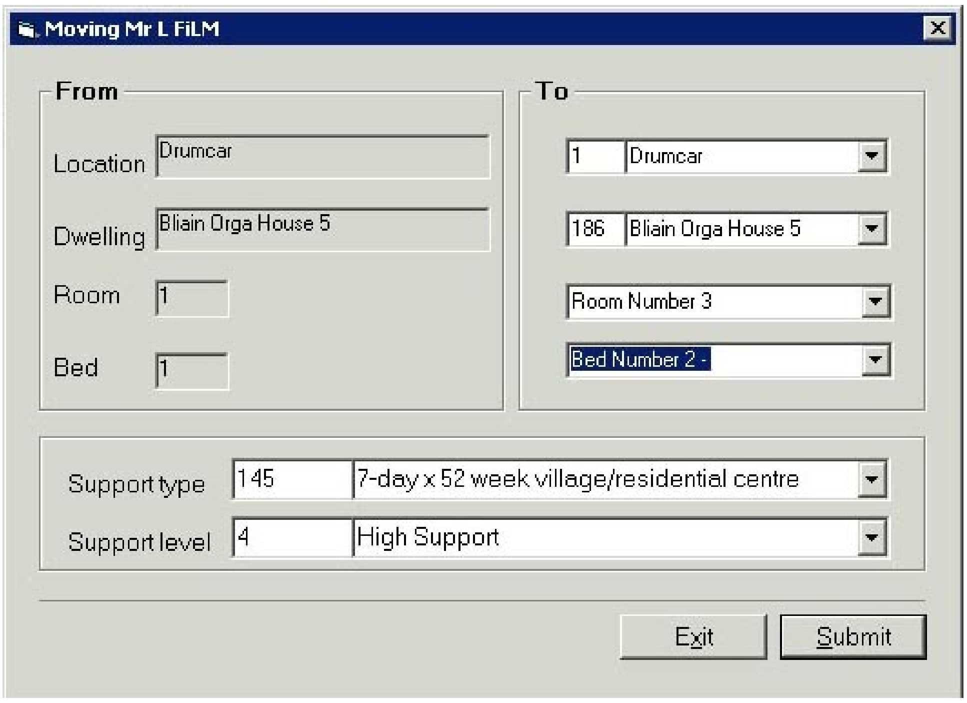
Room No. 6
Bed 1 Mr D MiDM

Room No. 7
Bed 1 Mr K DoKM

🟨 Reserved bed (Internal)
🟥 Reserved bed (External)

6.22 List of tasks to complete

1. Vacate bed and move client direct to an empty bed



Moving Mr L FiLM

From

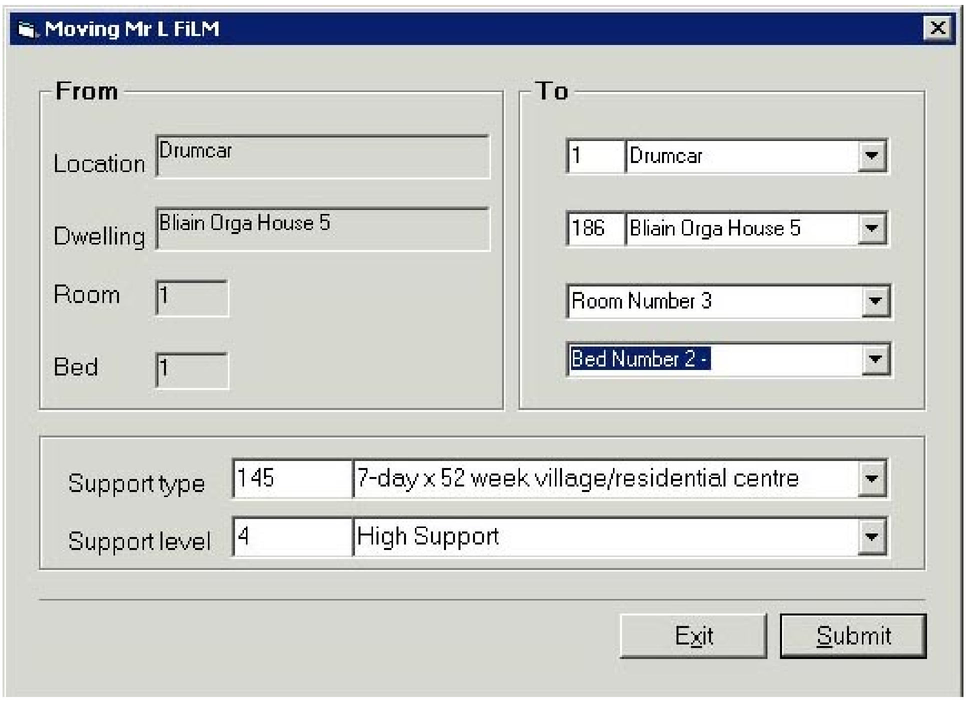
Location	Drumcar
Dwelling	Bliain Orga House 5
Room	1
Bed	1

To

1	Drumcar
186	Bliain Orga House 5
Room Number 3	
Bed Number 2 -	

Support type	145	7-day x 52 week village/residential centre
Support level	4	High Support

2. Reserve bed and move client direct to an empty bed



Moving Mr L FiLM

From

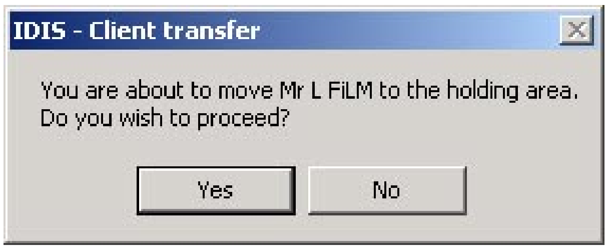
Location	Drumcar
Dwelling	Bliain Orga House 5
Room	1
Bed	1

To

1	Drumcar
186	Bliain Orga House 5
Room Number 3	
Bed Number 2 -	

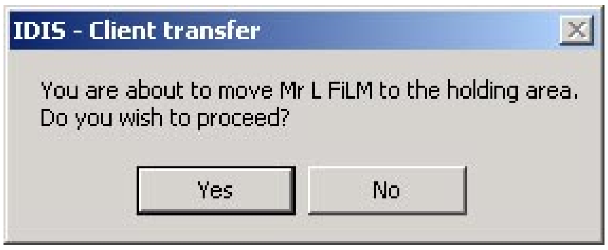
Support type	145	7-day x 52 week village/residential centre
Support level	4	High Support

3. Vacate bed and move client to holding area (internal transfer)


IDIS - Client transfer

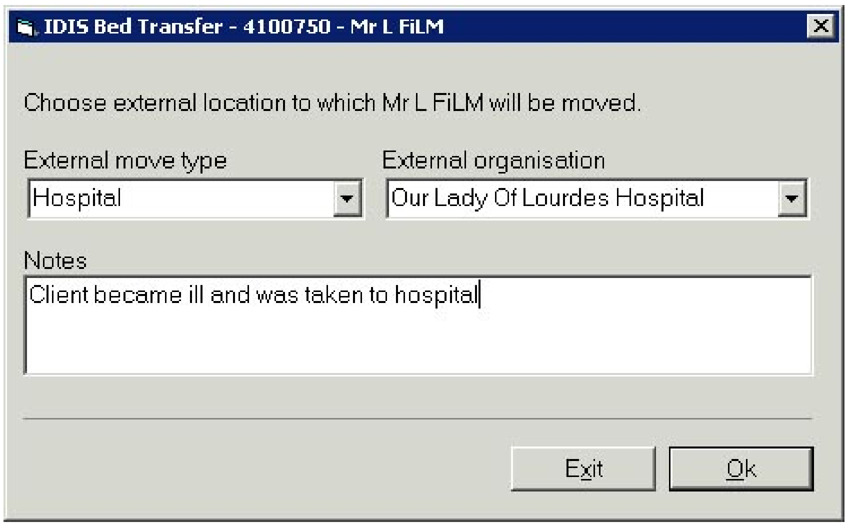
You are about to move Mr L FiLM to the holding area.
Do you wish to proceed?

4. Reserve bed and move client to holding area (internal transfer)


IDIS - Client transfer

You are about to move Mr L FiLM to the holding area.
Do you wish to proceed?

5. Reserve bed and move client to holding area (external move)


IDIS Bed Transfer - 4100750 - Mr L FiLM

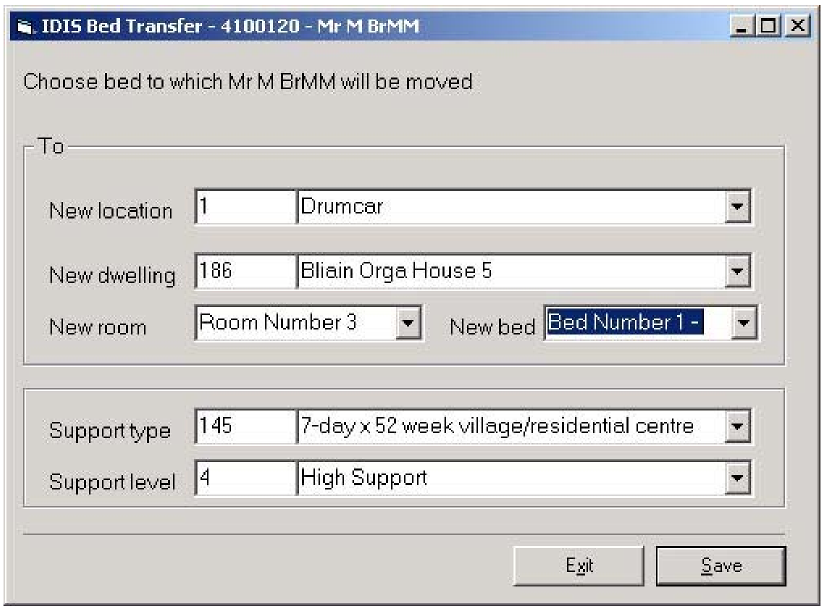
Choose external location to which Mr L FiLM will be moved.

External move type: Hospital

External organisation: Our Lady Of Lourdes Hospital

Notes: Client became ill and was taken to hospital

6. Move client from holding area to empty bed



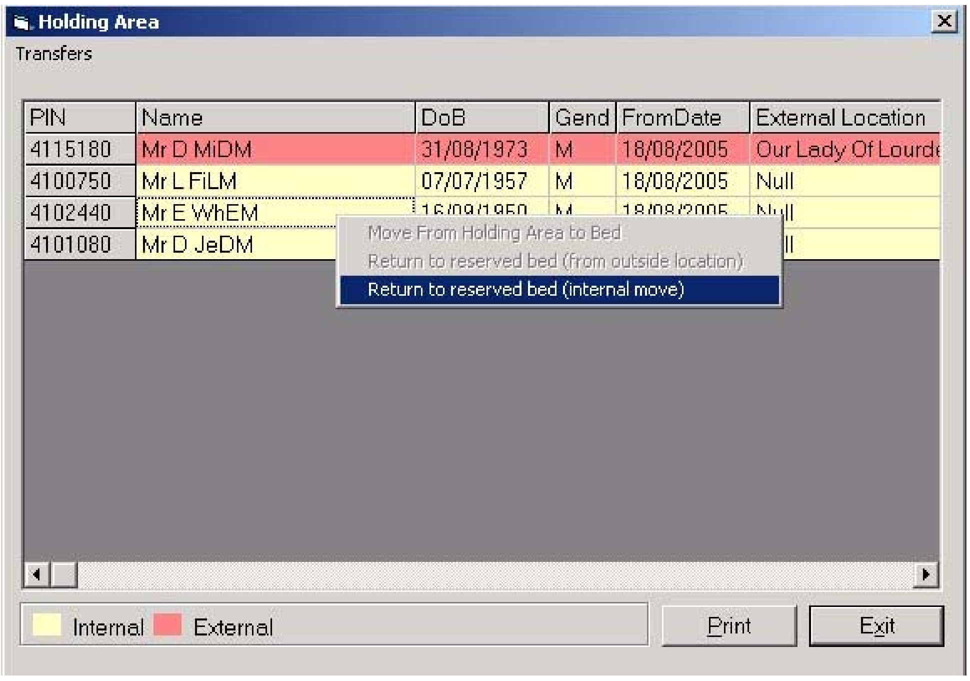
IDIS Bed Transfer - 4100120 - Mr M BrMM

Choose bed to which Mr M BrMM will be moved

To

New location	1	Drumcar
New dwelling	186	Bliain Orga House 5
New room	Room Number 3	New bed: Bed Number 1 -
Support type	145	7-day x 52 week village/residential centre
Support level	4	High Support

7. Return client to reserved bed (internal)



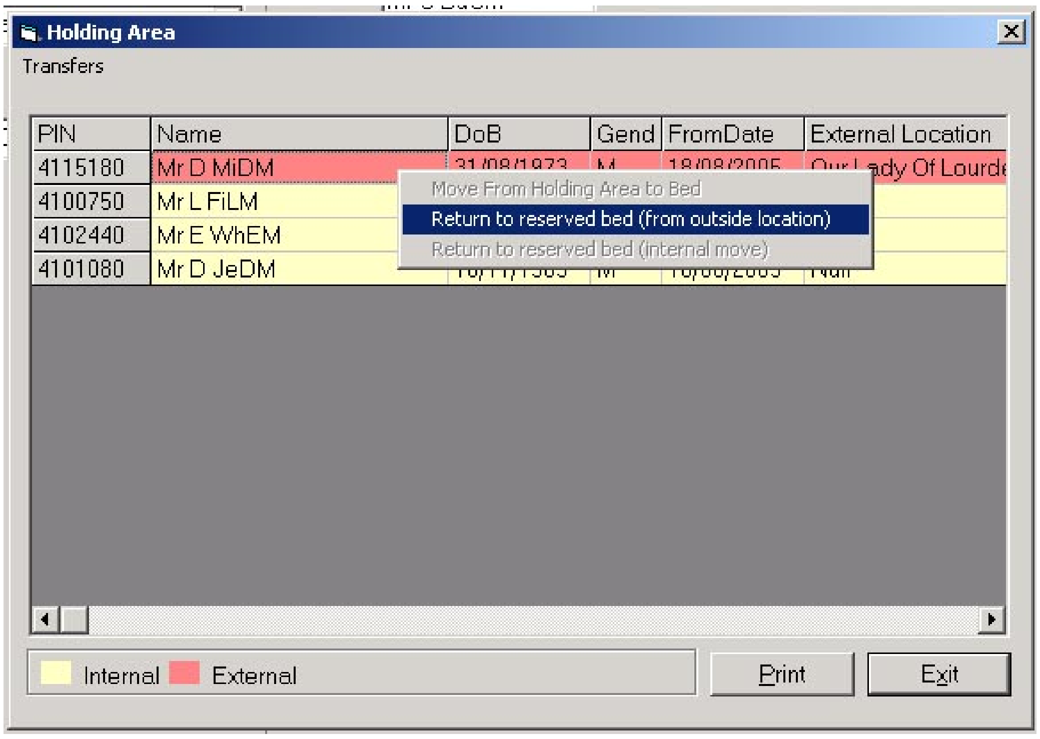
Holding Area

Transfers

PIN	Name	DoB	Gend	FromDate	External Location
4115180	Mr D MiDM	31/08/1973	M	18/08/2005	Our Lady Of Lourde
4100750	Mr L FiLM	07/07/1957	M	18/08/2005	Null
4102440	Mr E WhEM	16/09/1950	M	18/08/2005	Null
4101080	Mr D JeDM				ll

Move From Holding Area to Bed
Return to reserved bed (from outside location)
Return to reserved bed (internal move)
Internal
External
Print
Exit

8. Return client to reserved bed (external)


Holding Area

Transfers

PIN	Name	DoB	Gend	FromDate	External Location
4115180	Mr D MiDM	31/08/1973	M	18/08/2005	Our Lady Of Lourd
4100750	Mr L FiLM				
4102440	Mr E WhEM				
4101080	Mr D JeDM	16/11/1965	M	16/08/2005	Null

Move From Holding Area to Bed
Return to reserved bed (from outside location)
Return to reserved bed (internal move)
🟨 Internal 🟥 External
Print
Exit

6.23 Evaluator Feedback following heuristic review

Key: E = Issue raised by evaluator
U = Issue raised by key user
* = Issue addressed by developer

Below is a list of the tasks which the evaluator was asked to complete. The numbers under each task refer to the heuristic which was violated in each case. The 10 heuristics which are explained in section 4.33 are once again listed below.

1. Visibility of system status
2. Match the system to the real world
3. User control and freedom
4. Consistency and standards
5. Error Prevention
6. Minimise memory load – support recognition rather than recall.
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose and recover from errors
10. Help and documentation

Task #1 – Vacate bed and move client direct to an empty bed

1. Visibility of system status
   a. Caption on dialog ("Moving Mr X") should be more descriptive to reflect the specific action being taken. E*

2. Match between system and the real world
   a. Use of term "dwelling" does not reflect real world language. E

5. Error prevention
   a. User should not be able to select a room which contains no free beds. Assistance should be given to the user to help them locate and empty bed. UE*

6. Recognition rather than recall
   a. Once dialog is loaded, main bed management screen is hidden and modal screen can not be moved. User should be able to drag modal dialog away to allow viewing of underlying bed/room status. UE*

7. **Aesthetic and minimalist design**
   a. "Dwelling" dropdown box is not sufficiently wide to display full text of certain entries. This is a particular problem where entries are differentiated only by a number. UE*

Task #2 – Reserve bed and move client direct to an empty bed

1. **Visibility of system status**
   a. Caption on dialog ("Moving Mr. X") should be more descriptive to reflect the specific action being taken. E*
2. **Match between system and the real world**
   a. Use of term "dwelling" does not reflect real world language. E
5. **Error prevention**
   a. User should not be able to select a room which contains no free beds. Assistance should be given to the user to help them locate and empty bed. UE*
6. **Recognition rather than recall**
   a. Once dialog is loaded, main bed management screen is hidden and modal screen can not be moved. User should be able to drag modal dialog away to allow viewing of underlying bed/room status. UE*
7. **Aesthetic and minimalist design**
   a. "Dwelling" dropdown box is not sufficiently wide to display full text of certain entries. This is a particular problem where entries are differentiated only by a number. UE*
   b. "Support type" and "Support level" fields appear enabled but are disabled. Should be hidden from user or greyed out to indicate they are not to be used. UE*

Task #3 – Vacate bed and move client to holding area (internal transfer)

1. **Visibility of system status**
   a. Caption on message box should be more descriptive to reflect action being taken. E*
   b. Message box text makes no distinction between "vacate" and "reserve". E*
   c. Message box should contain graphic – vbQuestion E*

Task #4 – Reserve bed and move client to holding area (internal transfer)

1. **Visibility of system status**
   a. Caption on message box should be more descriptive to reflect action being taken. E*
   b. Message box text makes no distinction between "vacate" and "reserve". E*
   c. Message box should contain graphic – vbQuestion E*

Task #5 – Reserve bed and move client to holding area (external move)

1. **Visibility of system status**
   a. Caption on dialog should be more descriptive E*
   b. Confirmation message box should contain graphic – vbInformation E*
2. **Match between system and the real world**
   a. "External move type" caption should be changed to be more meaningful to users. E*

Task #6 – Move client from holding area to empty bed

1. **Visibility of system status**
   a. Caption on dialog should be more descriptive to reflect action being taken. E*
4. **Consistency and standards**
   a. Each field should only become visible as selection is made in preceding field. UE*
   b. Labels for each dropdown should be invisible where the dropdown is invisible. UE*
   c. Terminology should be consistent with other dialogs. "New" should be dropped from location, dwelling, room and bed labels. E*
   d. "Maximise" and "Minimise" buttons should be removed from the dialog to be consistent with other dialogs. E*

Task #7 - Return client to reserved bed (internal)

4. **Consistency and standards**
    a. Exit button is placed in bottom right corner of dialog – inconsistent with other screens. E*

Task #8 – Return client to reserved bed (external)

4. **Consistency and standards**
    a. Exit button is placed in bottom right corner of dialog – inconsistent with other screens. E*
8. **Aesthetic and minimalist design**
    a. Insufficient contrast in colours used to denote internal/external. U*

6.24 Conclusions

Following an evaluation of approximately two hours duration, the evaluator identified all but one of the issues which were raised by the key user during development. Furthermore, a number of the issues raised by the evaluator were not highlighted by the key user in her review of the interface during the development phase.

As can be seen from the key above, an "E" next to an issue denotes a usability issue raised by the evaluator, a "U" next to the issue denotes an issue raised by the key user and an asterisk next to the issue indicates that the issue was addressed and resolved by the developer prior to final implementation of the component.

For example, in task #1, the evaluator questioned the use of the term "Dwelling" deeming it to be in breach of second heuristic – "match between system and real world". As this was not raised by the key user in her evaluation, this terminology was not amended. In task #3, however, the evaluator pointed out that the message text was not sufficiently clear. While this was not observed by the key user, it was deemed by the developer to warrant amendment for reasons of clarity and consistency.

The only issue raised by the key user which was not raised by the evaluator was in relation to colour coding of items in the "holding area". As the key user is colour blind, she requested that the colours used to differentiate between "internal" and "external" records should be more stark in contrast.

In this simple example, the application of Nielsen's heuristics has proven to be quite successful and indeed it has been demonstrated that in conjunction with the task of identifying usability problems, a further task; determining which of issues, if not all, should be addressed by the developer. In the sample above, a certain degree of common sense was applied in choosing which usability issues should be acted upon by the developer.

Chapter 7: Conclusions and recommendations

7.1 Conclusions

The original purpose of this this dissertation was to attempt to develop a methodology which could be used to evaluate the usability of computer systems developed by the software development team in St John of God Services.

Having considered the structure and work practices of this department as well as the various resource constraints which impact on these work practices a methodology was sought which would suit the particular needs of the environment under study. A number of different methods and techniques were considered and their suitability to the domain under study was assessed.

Originally it was envisaged that the optimum technique would be one which involved the input of users but certain prohibitive factors outlined in chapter 5 deemed such an approach to be infeasible although it was acknowledged that user input into the iterative software design process would always be desireable where possible. It was concluded that the most appropriate method for use within St John of God Services was Nielsen's heuristic evaluation which could be conducted by staff within the software development department.

A small scale trial of this technique proved enlightening if inconclusive and it is clear that further trials, more extensive in their nature and involving multiple design iterations would be necessary to demonstrate the capabilities of the chosen technique in the domain under study.

Despite this, a number of positive conclusions can be drawn from the work conducted. In the sample case, all but one of the usability issues raised by the key user were raised by the evaluator during the heuristic evaluation. A number of these issues were resolved by the developer and others were left unchanged. Inherent in the heuristic evaluation process, therefore, is an onus on the developer to determine which among the issues raised should be addressed. Of the iussues raised by the evaluator which were not raised by the key user, most were fixed.

Regardless whether these issues would ever have been raised by a user of the component, they were, it was felt, in breach of the usability heuristics and were fixed to as to promote consistently high standards in usability of systems developed within St John of God Services.
Referring back to the introduction, usability is important for many differnet reasons. It impacts not only on users and how they perform their tasks but on software developers as it can mean the difference between success and failure of a system and on management as poor usability can adversely affect the productivity of the workforce to a level worse than without the system. (usabilityfirst.com). Also, the delicate relationship between the software development function and the departments which it supports, is another reason why as developers we should strive to produce software which enables users to perform their tasks in an effective and enjoyable way. It is the authors view that implementing software with poor usability has a negative influence on this relationship, damaging user confidence in IT systems and those who produce them and consequently can hamper efforts to attain staff buy in for future software projects.

It is the author's conclusion that while the methodology adopted in this study is far from ideal and is, in many ways, far removed from the recommended application of heuristic evaluation prescribed in the literature (useit.com), there are significant benefits to be gained by using it in the manner which the author has described.

7.2 Suggestions for further work

Among the literature reviewed in chapter 4, there appears to have been little work conducted which acknowledges the type of software development environment which exists in St John of God Services. Nielsen (1994) addresses this but in my view, does not go far enough, stopping short of suggesting an effective usability evaluation technique which can be applied in a software development environment of two software developers and with users who are rarely availble to contribute to the iterative design process. An attempt to measure the effectiveness of heuristic evaluation conducted by one evaluator would therefore be beneficial. Such a study would be required to measure evaluator feedback against user feedback over the duration of a software development project which required several iterations in the design cycle.

References

Ammenwerth E., Brender J., Nykanen P., Prokosch H. U., Rigby M., Talmon J. (2004) Visions and strategies to improve evaluation of health information systems. *International Journal of Medical Informatics*, vol. 73, pp. 479-491.

Ammenwerth E., Graber S., Herrmann G., Burkle T., Konig J. (2003) Evaluation of health information systems – problems and challenges. *International Journal of Medical Informatics,* vol. 71, pp. 125-35.

Barnum, C.M. (2002) *Usability testing and research.* New York: Pearson Education.

Baroudi, J.J. & Orlikowski, W.J. (1998). A short-form measure of User Information Satisfaction: A psychometric evaluation and notes on use. *Journal of Management Information Systems,* vol. 4, pp. 44-59.

Bias R. (1994). "The pluralistic walkthrough: Coordinated empathies". In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods, New York: Wiley. pp. 63-76.

Bowman D.A., Gabbard J.L., Hix D. (2002) A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence*, vol. 11, no. 4, pp. 404-424.

Carroll C., Marsden P., Soden P., Naylor E., New J., Dornan T. (2002) Involving users in the design and usability evaluation of a clinical decision support system. *Computer Methods and Programs in Biomedicine.* Vol 69, no. 2, pp.123-135.

Chin J.P., Diehl V.A., Norman K.L. (1998) Development of an instrument measuring user satisfaction of the human– computer interface. *Proc CHI*. pp.213–221.

Consolvo S., Arnstein L., Franza R.B. (2002). User Study Techniques in the Design and Evaluation of a Ubicomp Environment. *Proceedings of the 4th international conference on Ubiquitous Computing.*

Carroll, J.M. & Rosson, M.B. (1985). "Usability specifications as a tool in iterative development". In H.R. Hartson (Ed.), Advances in Human-Computer Interaction, Norwood. Ablex. pp. 1-28.

Demetriadis S., Karoulis A., Pombortsis A (1999). Graphical Jogthrough: expert based methodology for user interface evaluation, applied in the case of an educational simulation interface. *Computers and Education.* Vol. 32, no. 4, pp. 285-299.

Desurvire H.W., Kondziela J.M. & Attwood M.E. (1992). What is gained and lost when using methods other than empirical testing. *Proceedings of the HCI '92 Conference on People and Computers VII*. New York: Cambridge University Press. pp. 89-102.

Dumas, JS, and Redish, J (1999), A practical guide to usability testing, Intellect, Revised edition. (paper)

Dzida W., Herda S. and Itzfeldt W.D. (1978) User Perceived Quality of Interactive Systems, *IEEE Trans Softw Eng*. SE-4.4, pp. 270-276.

Dzida W., Wiethoff M., Arnold A.A. (1993). *ERGOguide: The Quality Assurance Guide to Ergonomic Software.* Delft University of Technology, Dept of Work and Organisational Psychology, PO Box 5050, 2600 GB Delft, the Netherlands.

Ferre X., Juristo N., Windl H., Constantine L. (2001) Usability basics for developers. IEEE Software vol. 18, no. 1, pp. 22-29

Fitzpatrick R. (1999). Strategies for Evaluating Software Usability. Department of Mathematics, Statistics and Computer Science, Dublin Institute of Technology, Ireland. pp. 1-9

Friedman C., Wyatt J.C. (1997) Evaluation Methods in Medical Informatics. New York: Springer. pp. 24-27.

Gediga G., Hamborg K.C., Duntsch I. Evaluation of Software Systems 2000
See – Gediga G., Hamborg K.C. and Düntsch, I. (2002) "Evaluation of Software Systems" In: A. Kent and J. G. Williams (Eds.) Encyclopaedia of Computer Science and Technology. Vol. 44 Marcel Dekker Incorporated pp. 162-196.

Gediga G., Hamborg K.C. and Düntsch I. (1999) The IsoMetrics Usability Inventory: An operationalisation of ISO 9241-10. *Behaviour and Information Technology*, vol. 18, no. 3, pp. 151-164.

Gray W., Salzman M. (1998). Damaged Merchandise? A Review of experiments That Compare Usability Evaluation Methods. *Human Computer Interaction*. Vol. 13, pp. 203-261.

Grief S. (1991). "The role of german work psychology in the design of artifacts". In J.M. Carroll (Ed.), *Designing Interaction. Psychology at the human interface* Cambridge University Press. pp. 203-226.

Hamborg K.C., Grief S. (1999). Heterarchische Aufgabenanalyse. In H. Dunckel (Ed.) *Handbuch psychologischer arbeitsanalyseverfahren*, Zurich. pp. 147-177.

Hamborg K.C., Vehse B., Bludau H.B. (2004) Questionnaire Based Usability Evaluation of Hospital Information Systems. *Electronic Journal of Information Systems Evaluation.* Vol. 7, Issue 1, pp. 21-30

Harrison B.L. (1991). Video annotation and multimedia interfaces: From theory to practice. In: *Proceedings of the Human Factor Society 35th Annual Meeting*, pp. 319-322.

Hendrickson G. & Kovner C.T. (1990). Effects of computers on nursing resource use: Do computers save time? *Computers in Nursing.* vol. 8, no. 1 pp. 16-22.

Murff H.J. and Kannry J. (2001) Physician satisfaction with two order entry systems. *Journal of the American Medical Informatics Association*. Vol. 8, no. 5, pp. 499-511.

Hix, D. & Hartson, H.R. (1993). Developing user interfaces: Ensuring usability through product and process. New York: Wiley

ISO 9241-10:1996 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 10: Dialogue principles.

ISO 9241-10:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.

Jeffries R. & Desurvire H. (1992) Usability Testing vs. Heuristic Evaluation: Was there a contest? *SIGCHI Bulletin* vol. 24, no. 4, pp. 39-41

Jimison H.B., Adler L.J., Coye M.J., Mulley A.G., and Eng T.R. (1999) Health Care Providers and Purchasers and Evaluation of Interactive Health Communication Applications. *American Journal of Preventive Medicine*. Vol. 16, no. 1, pp. 16-22.

John B.E. & Marks S.J. (1996) Tracking the Effectiveness of Usability Evaluation Methods. *Behaviour and Information Technology* vol. 16, no. 4/5, pp. 188-202.

Kaplan B., Shaw N. (2002) People, organisational and social issues: evaluation as an exemplar, in R.Haux, C.Kulikowski (Eds.) Yearbook of Medical Informatics 2002, Schattauer, Stuttgart, pp. 91-102

Karat J. (1997). "User-centered software evaluation methodologies". In M. Helender, T.K. Landauer & P. Prabhu (Eds.), Handbook of Human-Computer Interaction. Second Edition, Amsterdam: Elsevier. pp. 689-704.

Karat C.M., Campbell R., Fiegel T. (1992) Comparison of empirical testing and walkthrough methods in user interface evaluation. *Proceedings of the ACM CHI* 1992, pp. 397-404.

Kirakowski J. & Corbett M (1993). SUMI: The software usability measurement inventory. *British Journal of Educational Technology*. Vol. 24, pp. 210-212.

Jeffries R. & Desurvire H. (1992). Usability Testing vs. Heuristic Evaluation: Was there a contest? *SIGCHI Bulletin*, vol. 24, no. 4, pp. 39-41.

Kreitzberg, C. and Shneiderman, B. (1999) Making computer and internet usability a national priority, *Common Ground*. Revised version reprinted in Branaghan, R. J. (Editor), Design by People for People: Essays on Usability, Usability Professionals Assn, Chicago (2001), pp. 7-20.

Kushniruk, A.W. Evaluation in the design of information systems: applications of approaches form usability engineering. (2002) *Computers in Biology and Medicine* vol. 32, no. 3, pp. 141-149.

Kushniruk AW, Patel VL. (2004) Cognitive and usability engineering methods for the evaluation of clinical information systems. *Journal of Biomedical Informatics* vol. 37, no. 1, pp. 56-76.

Lewis C., Polson P., Wharton C. and Rieman J. (1990) "Testing a Walkthrough Methodology for Theory Based Design and Walk-Up-and-Use interfaces". *In proceedings of CHI 1990.* ACM, New York, pp. 235-242.

McConnell, S. (1996) Rapid development: taming wild software schedules. Microsoft Press: Redmond, Washington.

McDaniel, J.G. (2002) Improving system quality through software evaluation. *Computers in Biology and Medicine* vol. 32, pp. 127-140.

McKenna Sullivan J. (2004) Iterative Usability Testing in the Development of a Learning Technology System for Teaching Geographical Information Systems within a Civil Engineering Curriculum., *Unpublished MSC Thesis*, University of Missouri-Rolla.

Meister, D. (1986) Human Factors testing and Evaluation. Elsevier, Amsterdam.

Moehr J.R. (2002) Evaluation: salvation or nemesis of medical informatics? *Computers in Biology and Medicine* vol. 32, pp. 113-125.

Monk A., Wright P., Haber J., Davenport L (1993) Improving your human-computer interface: a practical technique. London: Prentice Hall (BCS Practitioner Series)

Nielsen J., Mack R.L. (1994) Usability Inspection Methods. New York: Wiley.

Nielsen J. (1994) Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. *http://www.useit.com/papers/*

Nielsen J. (1994) ‘Heuristic Evaluation’ in J. Nielsen and R.L. Mack (Eds.) Usability Inspection Methods, New York, Wiley. pp. 25-62.

Nielsen J. (1993) Usability Engineering. Boston: AP Professional

Nielsen, J. (1989) “Usability engineering at a discount”. In G. Salvendy & M. Smith (Eds.) Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Amsterdam: Elsevier. pp. 394-401

Nielsen, J. (1992) The usability lifecycle. *IEEE Computer*. Vol. 25, pp. 12-22

Patel V.L., Kushniruk A.W. (1998) “Interface design for health care environments: The role of cognitive science”. In *Proceedings of the 1998 AMIA Fall Symposium*; 1998 November 7–11. Orlando, FL.

Preece J., Rogers Y., Sharp H. (2002) Interaction design: beyond human-computer interaction. Wiley: New York

Preece J., Rogers Y., Sharp H., Benuyon D., Holland S., Carey T. (1994) Human Computer Interaction. Addison-Wesley.

Redmond-Pyle D. & Moore A. (1995) Graphical User Interface Design and Evaluation: a Practical Process. Prentice Hall International.

Rigby M. (2001) “Evaluation: 16 Powerful Reasons Why Not to Do It - and 6 Over-Riding Imperatives” in *Proceedings of the 10th World Congress on Medical Informatics (Medinfo 2001*), Patel V, Rogers R, Haux R (eds.), IOS Press: Amsterdam, pp. 198-202.

Rogers B., Hamblin C., Chaparro A. (2005) A Comparison of Two Evaluation Techniques for Technical Documentation. *Proceedings of the 13th International Symposium on Aviation Psychology*. pp. 1-4

Rowley D. and Rhoades D. (1992) The Cognitive Jogthrough: A Fast Paced User Interface Evaluation Procedure. *CHI 1992,* pp. 389-395

Rubin J. (1994). Handbook of Usability Testing. New York: Wiley.

Scriven, M (1967). “The methodology of evaluation”. In R. Tyler, Gagne & M.Scriven (Eds.), Perspectives of Curriculum Evaluation*,* Chicago: Rand McNally. pp. 39-83

Shakel B (1991). “Usability-Context framework, definition, design and evaluation”. In: B. Schakel & S.J. Richardson (Eds.), Human factors for informatics usability, Cambridge: Cambridge University, pp. 21-37

Shneiderman B. (1987). Designing the User Interface: Strategies for Effective Human Computer Interaction. 3rd Edition. Addison-Wesley: Massachusetts.

Sung Heum Lee (1999). Usability testing for developing effective interactive multimedia software: concepts, dimensions and procedures. *Educational Technoology and Society* vol. 2, no. 2, pp. 1-12.

Simeral E.J., Branaghan R.J. (1997) A Comparative Analysis of Heuristic and Usability Evaluation Methods. *Proceedings of the Society for Technical Communication annual conference.* pp. 307-309.

Sittig D.F., Kuperman G.J., Fiskio J. (1999) Evaluating physician satisfaction regarding user interactions with an electronic medical record system. *Proceedings of the American Medical Informatics Association Annual Symposium*. pp. 400–404.

Stead W.W. (1996). Matching the level of evaluation to a project’s stage of development. *Journal of the American Medical Informatics Association*. Vol. 3, no. 1, pp. 92-94.

Tyldesley D.A. (1988). Employing usability engineering in the development of office products. *Computer Journal*, vol. 31, pp. 431-436.

Virzi R.A., Sorce J.F. & Herbert L.B. (1993). “A comparison of three usability evaluation methods: Heuristic, think aloud and performance testing”. In *Designing for diversity: Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting 1993,* Santa Monica. Human Factors and Ergonomics Society. pp. 309-313

Wharton C., Jeffries R., Miller R. and Uyeda K.M. (1991). User Interface evaluation in the real world: a comparison of four techniques. In *Proceedings of ACM CHI '91 Conference on Human Factors in Computing Systems*, ACM, New York. pp. 119-124

Wharton, C., Rieman, J., Lewis, C., and Polson, P. (1994) "The Cognitive Walkthrough method: A practitioner's guide". In J. Nielsen and R.L. Mack (Eds.), Usability Inspection Methods, New York: John Wiley & Sons, pp. 105-141.

Whitefield, A., Wilson, F. & McDowell, J (1991). A framework for human factors evaluation. *Behaviour and Information Technology*, vol. 10, pp. 65-79

Wyatt C. (1997) "Evaluation of Clinical Information Systems" in Handbook of Medical Informatics, Van Bemmel J, Musen M (eds.), Springer-Verlag, Heidelberg, pp. 463-469.

Yamagishi, N. and Azuma, M. (1987). "Experiments on human–computer interaction evaluation", in G. Salvendy (ed.) Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems (Elsevier, Amsterdam) pp. 167-174

Zmud, R.W. & Boyton, A.C. (1991) "Survey measures and instruments in MIS: Inventory and appraisal". In K.L. Kraemer (ed.) The Information systems research challenge: Survey research methods. Boston MA: Harvard Business School. pp. 149-180.

Web References

Computerworld.com/developmenttopics/development/story/0,10801,71151,00.html
– last accessed 12th May 2005

Home.comcast.net/%7Etomtullis/publications/UPA2004TullisStetson.pdf
- last accessed 10th Apr 2005

Informit.com/guides/content.asp?g=it_management&seqNum=55&rl=1
- last accessed 12th Apr 2005

Ucc.ie/hfrg/questionnaires/sumi/
- last accessed 10th Feb 2005

Ucc.ie/hfrg/questionnaires/sumi/sumipapp.html
– last accessed 10th Feb 2005

Usabilityfirst.com/intro/index.txl
- last accessed 12th Jun 2005

Useit.com/papers/heuristic/learning_inspection.html
– last accessed 12th Apr 2005

www-sv.cict.fr/cotcos/pjs/MethodologicalApproaches/
evaluationmethods/evaluationpaperBannon.htm
- last accesed 8th Jun 2005