Summary and Review of the Suitability of Microsoft Access 2003 as a Software Prototyping Tool for a Small IT Department.

Samuel Sambasivam, Ph.D.
Azusa Pacific University,
Azusa, CA, USA
ssambasivam@apu.edu

Pat Levi
Nottingham, UK,
Europe
pat.levi@ntlworld.com

Abstract

A typical small IT department normally has many challenges placed upon it that need to be faced in order to be able to give a professional and effective service to the users (or customers) of the IT department. Common activities of a typical small IT department often include recording and planning work for the IT team, providing help desk support to users, and recording software changes as part of a change control process. This paper summarises the results of a project that evaluated a small sample of potential commercial software solutions before designing and building an evolutionary software prototype created with Microsoft Access 2003 as the recommended solution to satisfy the requirements agreed with the project sponsor to improve work planning, help desk support, and the recording of software changes in a small UK IT department. Before building the software prototype it was necessary to research topics related to building a software prototype with Microsoft Access 2003, it was also necessary to research any UK legal considerations that might apply to the environment in which the software prototype was implemented. The evolutionary software prototype was successful at satisfying all the requirements that were agreed with the project sponsor. This paper also reviews the suitability of Microsoft Access 2003 as a software prototyping tool based on the experience gained during the project through building a Microsoft Access 2003 software prototype to meet the requirements agreed with the project sponsor. The results of this review are expected to be of interest to other software developers who are considering using Microsoft Access 2003 as a tool to develop a software prototype.

Keywords: IT department work planning, IT help desk support, recording software changes, software prototype, Microsoft Access 2003, Access 2003 software prototype, help desk, human computer interaction, software engineering, UK IT legal considerations.

1. BACKGROUND OF THE PROBLEM

Project Overview

The first aim of the project was to conduct a high level evaluation of a small sample of commercial software solutions as well as to design and implement a bespoke purpose built software prototype built with Microsoft Access 2003 in order to recommend a potential software solution to improve work planning, help desk support, and the recording of software changes in a small IT department. The proposed solution had to satisfy the requirements that were discussed and agreed with the project sponsor in a formal requirements specification document. As there was not sufficient time within the project plan to learn both Microsoft Access 2003 and VBA it was initially planned to create the software prototype in Access 2003 without writing any VBA code, later in the project this strategy had to be changed.

At the start of the project 2nd author had no experience of using Microsoft Access 2003 and so 2nd author was unsure how well Microsoft Access 2003 would be suited to building a software prototype. 2nd author performed some preliminary research before starting the project and to my surprise found no information that reviewed the suitability of Microsoft Access 2003 as a software prototyping tool, this presented an opportunity to add a secondary aim to the project that could contribute to the world body of knowledge. The second aim of the project was to review the suitability of Microsoft Access 2003 as a software prototyping tool based on the experience gained during the project through building a Microsoft Access 2003 software prototype to meet the requirements agreed with the project sponsor. The review of the suitability of Microsoft Access 2003 as a software prototyping tool is expected to be of interest to other software developers who are considering to use Microsoft Access 2003 as a tool to develop a software prototype.

Problem to be addressed

The problem to be addressed was based on my experience of working as the IT Manager of a small IT department at a site within a UK company (the company is not a public authority) called Sogefi Filtration Ltd. Sogefi Filtration Ltd manufactures automotive filters for aftermarket and OE customers including Ford, Peugeot, Citroen and JCB and has an annual turnover of around £70 million in the UK, with around 933 employees located across three UK sites. The problem scenario was set in a small IT department in which each member of the IT department could receive work requests from users in an informal help desk environment. The help desk supported customers that were internal to the UK company and was not manned by dedicated staff, further consideration needed to be given to determine if this was a suitable policy. Without a suitable work planning and help desk solution IT staff could give poor service to users or fail to work on tasks based on the importance to the business, also new work requests from users could easily be forgotten or lost if they were not sensibly recorded at the time of being requested. IT help desks often have to deal with problems that have been resolved previously by other colleagues, if previous solutions could be easily identified then the help desk could be more efficient. Help desk work and scheduled IT work could result in the need to implement new software or make changes to existing software, for traceability it is useful to have a record of the software that has been implemented as a result of help desk work or scheduled IT work. The first problem to be addressed by the project was to recommend a potential solution to improve work planning, help desk support, and the recording of software changes in a small IT department that satisfied the requirements agreed with the project sponsor.

The second problem to be addressed by the project was to review the suitability of Microsoft Access 2003 as a prototyping tool based on the experience gained during the project through building a Microsoft Access 2003 software prototype to meet the requirements agreed with the project sponsor.

2. REQUIREMENTS, LITERATURE REVIEW, AND RECOMMENDED SOLUTION

List of requirements

A formal requirements specification was created to represent the list of requirements. Potential solutions included a small sample of COTS (Commercial Off The Shelf) solutions and a bespoke evolutionary software prototype written in Microsoft Access 2003. It was expected that the new system would be a bespoke evolutionary software prototype written in Microsoft Access 2003. The functional and non-functional requirements are summarised below in Table 1 and Table 2 of Appendix A.

3. LITERATURE REVIEW AND REVIEW OF COTS SOLUTIONS

The following literature review starts by investigating topics that are related to solving the project requirements by building a software prototype in Microsoft Access 2003. The literature review continues with a review of legal considerations for the project and afterwards reviews a sample of potential COTS solutions in order to evaluate the likelihood of being able to satisfy the project requirements with a COTS solution. After the literature review the chosen solution is confirmed.

Help desk

A help desk environment that allows user work requests to be recorded and tracked is one of the central themes in the project requirements, considerations needed to be made to determine the type of help desk environment that was required for the project. An IT help desk can be generally described as an on-demand service point that provides information or actions that assist help desk users in carrying out an IT related task. A dedicated help desk typically has the following three essential characteristics:

- The help desk is either centralised or consists or multiple help desks.
- Staff work exclusively for the help desk activities.
- Help desk staff may have either a basic knowledge with the ability to pass a problem on to more experienced staff or alternatively the help desk could be manned by experts.

The help desk is not a single subject discipline; typically the help desk is a combination of the disciplines of computing, information science, and service management (Marcella R. & Middleton I. 1996, pp.4-5). The use of help desk software on help desks is becoming common, typically two thirds of help desks use help desk software. The use of knowledge bases or artificial intelligence within help desks is still not common. The results of a British Library Research and Development Department funded project indicated that less than half of the help desks that were surveyed were manned by dedicated full time help desk staff, the majority of help desk staff provided help desk support whilst still carrying out their other duties (Marcella R. & Middleton I. 1996, p.7). As this project is focused on a small IT department in which all staff in the IT department were able to receive user work requests it was decided that the most cost effective solution was to man the help desk by staff who provide help desk support whilst carrying out their other duties.

There are two types of help desks depending on whether the clients of the help desk are internal or external to the organization (Heckman 1998, cited by Gonzalez et al. 2005 and Thomas 1996, cited by Gonzalez et al. 2005). The help desk for the project only had clients that were internal to the organisation therefore it was decided that the best solution was to continue using an internal help desk within the IT department of the organisation. Using an internal help desk is often beneficial as an internal help desk provides an important service to the organisation, it has been observed that an internal help desk can have a great impact on the productivity of an organisation since the help desk is resolving problems that may stop, delay, or otherwise impact the completion of daily business activities (Held 1992, cited by Gonzalez et al. 2005). The primary function of a help desk is problem solving of both new and previously solved problems. Solving previously solved problems is a form of knowledge acquisition (Gonzalez et al. 2005, p.393). Solving problems that have not been previously solved is known as knowledge creation (Gray 2001, cited by Gonzalez et al. 2005).

It has been suggested that it may be beneficial to deconstruct the process of users making requests and receiving support into four phases: Phase A is the greeting phase, during this phase someone or some thing asks the user "How may I help you?". Typically users may make a user request by telephone, by walk-up help desk or by electronic submission by email (Limoncelli, 1999, p.36). Phase B is the phase in which the user request is classified according to the type of request (such as a hardware printing problem) and a suitably skilled person is assigned the task of resolving the user request, afterwards the request is recorded and if the user request is to report a problem the problem symptoms are verified to see if the problem is repeatable (Limoncelli, 1999, pp.37-38). Phase C is the phase in which a solution to the user request is identified, planned and actioned (Limoncelli, 1999, p.38). Phase D is the verification phase, during this phase the user verifies that the request has been resolved (Limoncelli, 1999, p.40). Although the software prototype was not developed to match the four phases previously described (as this was not a specified requirement in the requirements specification) the prototype does have the capability of supporting the four phases.

Human Computer Interaction

Human computer interaction can be described as the study of how people interact with computers and the extent that computers are or are not developed for successful interaction with humans (Techtarget, 2005). Human computer interaction involves the use of a human computer interface, a human computer interface can be described as a software sub-system that mediates between the user and the program that transforms the computer into a tool for a specific application (Neelamkavil, F. & Mullarney, O. 1991, p.37). Human computer interaction typically includes consideration for how users form mental models about their interaction experiences, users often have different ways of learning and keeping knowledge and so individual users may have different cognitive styles (for example individual users may be classified as either left-brained or right-brained). Another factor that is often considered in human computer interaction is the impact of cultural and national differences (Techtarget, 2005).

It is advisable to consider human computer interaction requirements when designing a software prototype. Good user interface design is critical to the success of a system as a poorly designed interface may cause an otherwise good system to be rejected by the users of the system (Sommerville, 2001, p.328). The importance of a good human computer interface is also supported by Neelamkavil and Mullarney, according to Neelamkavil and Mullarney a human computer interface encompasses the aspects of a computer system that the user directly experiences therefore the performance of a human computer interface is a critical factor in the success or failure of application software (Neelamkavil, F. & Mullarney, O. 1991, p.37).

Human characteristics need to be taken into account when designing user interfaces, for example people have limited short term memory and they normally make mistakes when using a system (especially when stressed or trying to handle too much information). General user interface design principles should include user familiarity with the terms and concepts of the interface, consistency with how comparable operations are activated, minimal surprise in the behaviour of the system, recoverability mechanisms to recover from errors, user guidance in terms of meaningful feedback to allow users to recover from errors, and user diversity to allow tailored interaction facilities for different types of users (Sommerville, 2001, p.330).

It is useful to know how to evaluate a human computer interface, such skills are often useful when creating a user interface or when reviewing COTS solutions that contain user interfaces. Interface evaluation involves assessing the usability of an interface and the suitability of the interface to meet the user requirements. Metrics for the usability of an interface might include learnability (how long does it take a new user to be productive with the interface), speed of operation (is the interface speed of operation sufficient to satisfy the user's working practice) robustness (for example how tolerant is the interface of user error), recoverability (for example can the system recover from user errors), and adaptability (is the interface adaptable or is it closely tied to a single model of work) (Sommerville, 2001, p345). According to Schneiderman the principles of human computer interface design incorporate eight golden rules as follows:

- Strive for consistency (for example keep terminology consistent, also consistent sequences of actions should be required in similar situations)
- Enable frequent users to use shortcuts (as users become more familiar with an interface it is desirable to reduce the number of interactions)
- Offer informative feedback (for every user action there should be some informative feedback)
- Design dialog to yield closure (sequences of actions should be designed with a beginning, middle, and end)
- Offer simple error handling (offer simple mechanisms for handling errors)
- Permit easy reversal of actions (allow the user to reverse actions)
- Support internal locus of control (design the interface to make users the initiators of actions rather than the responders)
- Reduce short-term memory load (limitations of human processing in short term memory should be considered when de-

signing interfaces, for example short-term memory load can be reduced by keeping displays simple and using pull down menu's and icons) (San Jose State University, 2005).

Another important factor to consider when designing or evaluating user interfaces is the use of colour, general guidelines for the use of colour in user interfaces include:

- Limiting the number of colours used, for example no more than five colours should be used in a window and no more than seven colours should be used in a system interface.
- Use colour to show a significant change in system status.
- Use colour coding to aid the task that users are trying to perform, for example if the user is looking for anomalies that have been identified by the system highlight the anomalies in a particular colour.
- Use colour coding in a consistent way, for example display all error messages in the same colour.
- Be careful with colour pairings, for example the human eye can not focus on red and blue simultaneously.
- Do not use colour to represent meaning, for example a vehicle driver usually interprets red as danger whilst a chemist usually interprets red as hot (Sommerville, 2001, p339).

Microsoft Access

As the most likely solution to the project was a Microsoft Access 2003 software prototype certain considerations needed to be made regarding the use of Microsoft Access 2003 and the potential impact on the project.

The first consideration to be made was to ensure that Microsoft Access has the capability of creating a multi-user relational database as this was a project requirement. A Microsoft Access desktop database has a file extension of .mdb and is a fully functioning RDBMS (Relational Database Management System), a Microsoft Access desktop database (identified by the .mdb file extension) can be either a stand alone RDBMS on a single PC or a shared multi-user client server database when used on a network (Viescas, 2003, p.6).

Microsoft Access implementation architectures needed to be considered to determine the most suitable implementation architecture to match the project requirements with the available resources. Investigating Microsoft implementation architectures also helped to identify potential solutions for coping with future growth of the application. Example Microsoft Access 2003 implementation architectures include:

- A Microsoft Access desktop database (.mdb file) can be installed as a stand alone database on a single PC (where both the client and server exist in the same PC). Microsoft Access is used as the RDBMS when the Microsoft Access desktop database is used to build an application.
- Two-tier client server system: A Microsoft Access desktop database (.mdb file) can be installed on a network server. A two-tier client server system normally has an application user interface on the client computer and the corresponding application database stored on a server computer, the client computer requests services directly from the server computer via the application user interface. The actual application logic can run on either the client or the server (Webopedia, 2005b).
- Two-tier client server system: A Microsoft Access data-only desktop database (.mdb file) can be installed on a network file server with linked tables over the network into multiple Microsoft Access desktop databases.
- Two-tier client server system: A Microsoft Access data-only desktop database (.mdb file) can be placed in a Microsoft SQL server database on a network server with linked tables over the network into multiple Microsoft Access desktop databases.
- Two-tier client server system: The database can be designed in Microsoft SQL Server and connected over a network to multiple Microsoft Access project files (.adp) running on different client PC's. Microsoft SQL Server Desktop Engine is used as the RDBMS when a Microsoft Access project file is created.
- Three-tier client server system: Web browsers act as the client interface to a Web server, the Web server connects to a Microsoft Access database (.mdb file) on a file server (Viescas, 2003, p.6, p.52). A three-tier client server system has an application user interface on the client computer, a middle tier application

server for the application logic that processes the application data, and a third tier database management system server that stores the data required by the middle tier application (Webopedia, 2005c).

Due to limited resources it was expected that the software prototype would have to be implemented as a Microsoft Access desktop database on a single PC (where both the client and server exist in the same PC), however, providing that the necessary resources were available the completed software prototype would be still be capable of being implemented as a multi-user two-tier client server system on a network. A Microsoft Access desktop database can be used as a multi-user two-tier client server system on a network by placing the database (.mdb file) on a Windows network server (such as a server running Microsoft Windows Server 2003) and allowing shared network access from clients connected to the network that are running Microsoft Access. There are performance benefits in converting a multi-user two-tier client server Microsoft Access desktop database on a network into a Microsoft Access data-only desktop database, the main performance benefit is reduced network traffic. Microsoft Access data-only desktop databases are able to reduce network traffic by only requiring the Microsoft Access tables to be accessed over the network, each client will have a local copy of the application logic (in Microsoft Access the application logic is comprised of Microsoft Access queries, forms, reports, data access pages, macros, and modules). The Microsoft Access database splitter could be used to convert the software prototype from a Microsoft Access desktop database into Microsoft Access data-only desktop database (Viescas, 2003, p.1204). If required the Microsoft Access database upsizing wizard can be used to upsize a Microsoft Access desktop database into an alternative two-tier client server solution that uses Microsoft SQL server. Reasons to upsize to a two-tier client server solution that uses Microsoft SQL server include:

- The number of concurrent users needs to exceed 20 users.
- The database is rapidly growing in size and will soon exceed 100 MB.
- Users are complaining about the performance of the database and the problems have not been able to be resolved by making performance related modifications to the application design (Viescas, 2003, p.1135).

Although options exist to upsize the Microsoft Access database (as described above) that was used for the software prototype the initial size of the Microsoft Access database was not considered to be a problem for the project as the maximum size of a single Microsoft Access 2003 database is 2GB, also if required several Microsoft Access databases (each not larger than 2GB) can be attached to the application database that contains the forms, reports, macros, and modules to allow a Microsoft Access 2003 application to access more than 2GB of data (Viescas, 2003, p.137).

Microsoft Access database housekeeping considerations needed to be made to determine how to keep the database in good working order. The Microsoft Jet Database Engine treats a Microsoft Access database file as a series of 4096 byte blocks, the set of records in a Microsoft Jet table are stored in a series of blocks. Each time records are added or deleted the table blocks become fragmented. Compacting a Jet database de-fragments the blocks and improves read/write performance to the table. Compacting the database also improves the performance of Microsoft Access database indexes, updates table statistics stored in the database, and allows the Microsoft Jet Engine to re-optimise stored queries in the database (Microsoft Help And Support, 2005a). To compact a Microsoft Access 2003 database the compact and repair database utility needs to be used. The compact and repair database utility in Microsoft Access 2003 attempts to repair corruption in tables, forms, reports, and modules and compacts the database on disk. The compact and repair database utility can be set to run automatically when the database is closed by selecting the following options after opening the Microsoft Access 2003 database: Tools > Options > General tab > select the compact on close check box and click ok (Online Training Solutions Inc, 2003, pp.218-219).

A multi-user Microsoft Access database needs to be configured correctly for the database to manage database record locking. For record locking Microsoft Access 2003 allows pessimistic locking and optimistic locking. Pessimistic locking locks a record for the full duration in which it is being edited whilst optimistic locking only locks the record for the brief time that it is being changed. To share an Access database on a LAN each client workstation must have a copy of Access installed (Online Training Solutions Inc, 2003, p.231). For the project it was decided that the safest option was to use pessimistic locking, this can be set by selecting the following options after opening the Microsoft Access database: Tools > Options > Advanced, set default mode to shared, set default record locking to edited record, and make sure that the open database using record-level locking check box is selected (Online Training Solutions Inc, 2003, pp.232-233).

Recording software changes

Having the ability to record software changes is a common part of a change management process. During the lifetime of a system changes are to be expected as organisational needs and requirements change during the lifetime of a system (Sommerville, 2001, p.647).

Software process model

A suitable software process model needed to be chosen for the project. Generic software process models include the waterfall model, evolutionary development, formal systems development, and reuse-based development. Formal systems development is based on producing a formal mathematical system specification to produce a program, processes based on this model are only used in a few organisations. Reuse-based development is based on the existence and use of reusable components that are integrated into a system instead of building a system from scratch. The waterfall model (also known as the software life cycle) consists of phases that cascade from one phase to another with the next phase not starting until the previous phase has finished. The main stages of the waterfall method are requirements analysis and definition, system and software design, implementation and testing, and operation and maintenance. The partitioning into distinct stages requires commitments to be made in the early stage of the software process therefore the waterfall model does not respond well to changing requirements and so is best suited when the requirements are well understood (Sommerville, 2001, pp.44-48). Evolutionary development involves developing an initial implementation from abstract specifications that is refined with input from the customer into a system that satisfies the customer requirements. Evolutionary development performs the activities of specification, development, and validation concurrently. The two types of evolutionary development are exploratory development and throw-away prototyping. Exploratory development focuses on the parts of the system that are best understood and evolves by adding new features suggested by the customer, the overall objective is to deliver a final working system. Throw-away prototyping aims to gain a better understanding of the customer requirements in order to develop a better requirements definition for the system. A throw-away prototype concentrates on the customer requirements that are least understood, as the name implies a throw-away prototype is designed to be discarded after use and as such should not be used as a final working system (Sommerville, 2001, pp.44-48). For small to medium systems (up to 500,000 lines of code) with a fairly short lifetime the evolutionary approach is well suited, however for large systems or systems with a long lifetime a hybrid process that incorporates the best features of the waterfall model and the evolutionary development models is recommended (for example parts of the system that are well understood could be specified and developed using the waterfall model whilst less understood parts of the system could be specified and developed using exploratory development or by using a throw-away prototype to better understand the requirements after which the waterfall model could be used) (Sommerville, 2001, pp.44-48).

It was decided that the project would use a hybrid process that uses the waterfall model through out the project with evolutionary development for the software prototype. As the project is split into distinct stages that includes activities such as producing a project specification, producing a project design, developing and implementing a software prototype, system evaluation and testing etc then the waterfall model fits the overall project, however, as the project also involves producing a evolutionary software prototype that is intended to be retained as the final system then the evolutionary development model also applies. The availability of a requirements specification due to the use of the waterfall model was able to assist the evolution of the software prototype as there was a requirements specification that could be referred to when building and evaluating the software prototype. A system with a long lifetime is suited to the hybrid process involving both the waterfall and evolutionary development model. As the lifetime of the system for the project was not expected to be short this further confirmed the suitability of using a hybrid process that uses the waterfall model throughout the project with evolutionary development for the software prototype.

Software design methodology

A software design methodology needed to be chosen for the project that was suitable for building a software prototype using Microsoft Access 2003. Microsoft Access 2003 creates relational databases and supports the use of objects, the following object types can be used in Microsoft Access 2003: tables, queries, forms, reports, pages, macros, and modules. The object type called table is used to store information, the other object types are used to manage, manipulate, analyse, retrieve, display, or publish information contained in tables (Online Training Solutions Inc, 2003, p.3). Microsoft Access can use VBA (Visual Basic for Applications) code, VBA is not a true object-oriented programming language (OOPL) due its lack of object-oriented programming features such as inheritance (Ondotnet, 2005).

As Microsoft Access supports the use of objects OOD (Object Oriented Design) was proposed as the main software design methodology to be used to design the software prototype. The following items needed to be considered in order to determine if OOD alone would be likely to satisfy the project requirements :

- A Microsoft Access relational database needed to be designed for the software prototype and OOD is not normally well suited to designing a relational database.
- Although Microsoft Access uses objects Microsoft Access was not designed to be a full object-oriented programming environment (Viescas, 2003, p.397).
- Object oriented languages are well suited to OOD. Although Microsoft Access uses objects neither Microsoft Access nor VBA (VBA can be used by Microsoft Access) are true object oriented programming languages as features such as object inheritance are not supported.

Based on the above information it was decided that the design of the software prototype should avoid OOD models that are based on object inheritance (such as object class hierarchy diagrams as this type of diagram includes representation for object inheritance).

Two of the main models for modular decomposition are an object-oriented model and a data flow model (also known as a data flow diagram or DFD). In an object-oriented model the system is decomposed into a set of communicating objects whilst in a data flow model the system is decomposed into functional modules that receive data input and process the data and produce output (Sommerville, 2001, pp.229-232). A typical architectural design process decomposes elements into more detailed elements, the decomposition process continues until the architectural structure is sufficiently fine-grained to be assigned to software development teams to design and implement each piece of the architecture (McGregor J. 2004, pp.66-67).

It was decided that a hybrid software design methodology using selective functional models and selective object-oriented design models (selective OOD design models were used that are not based on object inheritance as neither Microsoft Access nor VBA are true object-oriented programming languages as features such as object inheritance are not supported) would be the most suitable solution for designing the Microsoft Access software prototype. The data flow diagrams (also known as data flow models) and specific requirements specifications using standard forms that were produced in the formal requirements specification document were able to be referred to in the design process to provide functional decomposition. Modular decomposition was also used in the software design process as the chosen hybrid design methodology uses object-oriented models as well as the data flow models that were created in the requirements specification document. It was decided that an entity relationship diagram (also known as an entity relationship model) supplemented by a data dictionary would be suitable to design the database for the project.

Software prototyping

The use of software prototyping to develop software solutions has increased in popularity over the last 25 years. In the early 1980's organisations used prototyping in approximately thirty percent of development projects, by the early 1990's the use of prototyping doubled to approximately sixty percent of development projects (McClendon et al. 1996, cited by Wikipedia, 2005). User interfaces are best to be developed using prototyping as it is not usually possible to specify user interfaces effectively with a static model. Evolutionary prototyping with end-user involvement is the most suitable way to develop graphical user interfaces for software systems (Sommerville, 2001, pp.188-189). In a study of information systems managers and other information systems professionals at 112 different organisations it was found that information systems software cost estimating was an important concern. It was reported that the completion of only one of every four systems development projects were within their estimated costs (Lederer A. & Prasad J. 1995, p.125). Evolutionary prototypes allow small and medium sized systems to be rapidly developed and delivered, normally this results in system development costs being reduced (Sommerville, 2001, p.178). An evolutionary software prototype would be a suitable type of prototype to solve the project requirements for the following three reasons: The use of an evolutionary software prototype is suitable as most of the requirements are well understood (a throw-away prototype is not required for requirements elicitation). The use of an evolutionary software prototype is likely to reduce system development costs. The use of an evolutionary software prototype can assist the development of the graphical user interface.

UK legal considerations

The target implementation environment for the project was a small IT department that is internal to a UK organisation, the organisation is not a public authority. A review was conducted into UK legal considerations that may be applicable to the project, the review included the Data Protection Act 1998, Freedom Of Information Act 2000, Privacy and Electronic Communications Regulations 2003, as well as other legal considerations (the review was not an exhaustive review of all UK law, however the review did cover key legislation that might be applicable to the project). The Data Protection Act (1998) was the most likely legislation to be applicable to the project. In the UK the Information Commissioner and his staff are responsible for ensuring that organisations that are processing data are doing so in line with the obligations that are placed upon them by the various pieces of legislation such as the Data Protection Act 1998, Freedom of Information Act 2000, and the Privacy and Electronic Communications Regulations 2003 (Information Commissioner, 2005e).

The Data Protection Act 1998 aims to balance the rights of individuals with legitimate reasons for using personal information. Individuals are given certain rights about the information held about them that places obligations on data controllers who process the personal information. Personal information includes both facts and opinions about an individual. Data controllers must notify the Information Commissioner's Office if they are processing personal information (unless their processing is exempt), notification costs £35 per year. Data controllers who process personal data must comply with the following eight rules of good practice:

- Data must be fairly and lawfully processed.
- Data must be processed for limited purposes.
- Data must be adequate, relevant and not excessive.
- Data must be accurate and up to date.
- Data must be not kept longer than necessary.
- Data must be processed in accordance with the individual's rights.
- Data must be secure.
- Data must not be transferred to countries outside the European Economic area unless the country has adequate protection for the individual (Information Commissioner, 2005a).

In the Data Protection Act 1998 personal data means data that relates to a living individual who can be identified either by those data or from those data together with any other information held by the data controller

or likely to come into the possession of the data controller (HMSO, 2005). Holding information about others in a domestic environment for family, household, or personal reasons is exempt from the Data Protection Act. If you hold information about other people for non-domestic purposes then the Data Protection Act will impose legal obligations that must be complied with. A data controller is the person (individual, company or organisation) who decides why personal data is held and the way in which personal data is dealt with. Examples of personal data includes collecting information about living individual customers in terms of where they shop, how they pay for goods, and the delivery address. Examples of information about others that does not count as personal data includes information about deceased individuals or information about companies (Information Commissioner, 2005f).
For the purposes of the project only very general personal information required to be held about living individuals such as: First Name, Last Name, Department, and Site. The Information Commissioner's Office produces a notification exemptions self assessment guide for data controllers to clarify if they need to declare (known as notification) personal data under the Data Protection Act 1998. According to the notification exemptions self assessment guide notification is not required if the processing of personal data is for staff administration, staff administration includes work management (Information Commissioner, 2005b). As the Work Request Database (software prototype) is only processing personal data for staff work management purposes notification should not be required. If any further clarification is required then the Information Commissioner’s Office helpline could be contacted, the Information Commissioner’s Office helpline telephone number can be found at the end of the Data Protection Act Factsheet (Information Commissioner, 2005b).

The Freedom Of Information Act 2000 enables access to information held by public authorities by public schemes or by general rights of access. To comply with access to information through public schemes each public authority must routinely make information available to the public. To comply with general right of access individuals can request access to information held by a public authority (some exemptions apply in the interest of public interest) and the public authority has to respond to the request for information (Information Commissioner, 2005d). The Freedom Of Information Act 2000 applies to public authorities in England, Northern Ireland, and Wales. Examples of public authorities include government departments, local authorities, hospitals, medical surgeries, dentist surgeries, schools, universities, police forces and prison services. Examples of recorded information covered by the Act include emails, meeting minutes, research and reports (Information Commissioner, 2005c). The target implementation environment for the project was not a public authority therefore the Freedom Of Information Act 2000 does not apply to the project.

Review of commercial software solutions

An evaluation of a small sample of potential commercial software solutions was conducted to determine if the system could be satisfied by a COTS (Commercial Off The Shelf) solution. Three COTS solutions were reviewed to determine how suitable each COTS solution was for satisfying the requirements of the project.

- Track-It! 6.5 Professional Edition is an IT asset management system that includes help desk functionality. A trial version of Track-It! 6.5 Professional Edition was downloaded and installed for evaluation purposes. Track-It! calls a user request a works order. Works order data includes: requester (requested by user name), location (site), workstation ID (not mandatory), task summary, task type, task priority, assigned technician (also a works order can be assigned to a group such as a group of technicians), date assigned, due date, completed date, and solution text. When adding a new work order a knowledge base of previous problems (works orders) can be searched by specifying selection criteria for the work order summary text of previous works orders. Also previous works orders (user work requests) for a particular user (or asset) can be reviewed when adding a new works order. A small IT department typically may manage over a hundred workstations, Track-It! 6.5 Professional Edition is the most suitable product in the range of Track-It!

products as this product is designed for managing hundreds of workstations (Intuit, 2005). Track-It! provided good functionality for user work requests within a help desk environment but no functionality for recording software changes and limited functionality for work planning.

- A trial version of Request Tracker (version 3.1) was downloaded and installed for evaluation purposes. Request Tracker allows customer requests (user work requests) to be recorded and progressed in a help desk database. The software tracks who requested what and when they requested it, what was done to resolve the request, who handled the request and how much time was spent to resolve the request. One screen records all the information about a single request (Cniche, 2005a). Request Tracker does not seem to use consistent terminology for a user work request, for example a user work request is referred to as either a work order or a request depending on which function is being used (for example one screen is called find request and another screen is called print work order). Request data includes: customer (requested by user name), department/company name (the customer department name or customer company name, the same attribute is used to hold either type of information), category (categories of request might include hardware, software etc), entered by (the name of the IT staff member who recorded the request), request short description, request number, request date, due date, request description (detailed description of the request), assigned to (the name of the IT staff member that the request is assigned to), request priority, resolved date, resolved status (displayed as a tick when the request is resolved), response (detailed description of how the request was resolved), and the hours and minutes spent on the request (optional use if the customer is to be charged). Request Tracker provides two methods to locate previous customer requests; either by applying a filter or by performing a find. Request Tracker provided good functionality for user work requests within a help desk environment and reasonable functionality for work planning but no functionality for recording software changes. Some problems were experienced with the custom report writer (a custom report could only be sorted on a maximum of two fields, also when trying to run a custom report sorted on two fields the report crashed with an error message regarding the number of parameters).
- A Windows compatible trial version of Surround SCM (version 3.1.3) was downloaded and installed for evaluation purposes. Surround SCM is a software change management tool for controlling software source code and other digital assets. The Surround SCM file detail pane can be used to view the properties of selected files including the history of changes to each file and the availability of each source file to be checked out of the source code repository for modifications to be applied. Surround SCM is a client server solution that is available for several operating systems including Windows, Solaris, Linux, and Mac OS X. The software can integrate with several software development tools including Visual Studio .NET, Visual C++, Visual Basic, Borland JBuilder, Delphi, Dreamweaver MX, WebSphere, CodeWarrior, Eclipse, Ant, NAnt, CruiseControl, IntelliJ IDEA, and Visual Build Professional. The software is licenced by either named users or concurrent user licences (or a combination of both) (Seapine, 2005a). During the trial of Surround SCM source code was checked out of the source code repository and modified, once checked back in to the source code repository each source code version number was automatically incremented to the next version number. A change history of each source code file could be displayed. Differences between versions of source code could be identified with the comparison results displayed side by side in the same panel. Surround SCM provided extensive functionality to manage and control software source code, however, no functionality was evident during the trial of the software for recording implementation details. Surround SCM provided no functionality for work planning or help desk support.

The estimated cost of each COTS solution is presented below in Figure 1, (Appendix A) each COTS solution cost includes provision for 5 days consultancy at £995 per day (totalling £4,975) for product installation services and training.

Figure 2 (Appendix A) below summarises the suitability of each COTS solution to satisfy the requirements of the project. The evaluation of the suitability of each COTS solution is based mainly on the functional requirements (the only non-functional requirement included is the requirement for the solution to be capable of being a multi-user solution that is can have up to 5 concurrent users).

- As shown above in Figure 2 Track-It! 6.5 Professional Edition obtained a weighted level of fit score of 52 out of 100, this score indicated that Track-It! 6.5 Professional Edition had a low likelihood of satisfying the project requirements.
- As shown above in Figure 2 Request Tracker (version 3.1) obtained a weighted level of fit score of 51 out of 100, this score indicated that Request Tracker (version 3.1) had a low likelihood of satisfying the project requirements.
- As shown above in Figure 2 Surround SCM (version 3.1.3) obtained a weighted level of fit score of 22 out of 100, this score indicated that Surround SCM (version 3.1.3) did not satisfy many of the system requirements. Surround SCM (version 3.1.3) had a very low likelihood for satisfying the project requirements.

Recommended solution

Failure to find suitable COTS solutions confirmed the need to write bespoke software to solve the project requirements. As most of the system requirements were understood an evolutionary software prototype was chosen as the solution to solve the project requirements. The evolutionary software prototype would only include the most important and best understood system requirements, the requirements to include in the software prototype were agreed with the project sponsor. A comparison of the estimated cost of the solutions considered can be seen below in Figure 3 (Appendix A).

In order to have a fair comparison between the expected suitability of the evolutionary software prototype and the suitability of each of the three COTS solutions the same evaluation criteria that was used to evaluate the suitability of each COTS solution was used to evaluate the expected suitability of the evolutionary software prototype. A comparison of the expected suitability of the evolutionary software prototype with other solutions considered can be below in Figure 4 (Appendix A).

The expected total weighted level of fit score for the requirements that were agreed with the project sponsor to be included in the evolutionary software prototype is 80 out of 80, this score indicates that the evolutionary software prototype has a high likelihood of success for satisfying all the project requirements that were agreed with the project sponsor for inclusion within the evolutionary software prototype. As shown above in Figure 4 the expected total weighted level of fit score of the evolutionary software prototype for all the project requirements (including some of the requirements that were agreed to be excluded from the evolutionary software prototype) is 90 out of 100, this score indicates that the evolutionary software prototype is likely to satisfy most of the project requirements and offers the best possibility of success out of the options that were considered. Although not included within the project (due to time constraints), the evolutionary software prototype has the potential to solve all the project requirements by further evolving the evolutionary software prototype after the initial project has been completed.

The necessary skills and resources needed to be available to build the evolutionary software prototype. The main resources required for developing the evolutionary software prototype was identified as a PC running Windows XP (Home Edition), software to develop a software prototype (only Microsoft Access 2003 was available), a learning source to learn how to use Microsoft Access 2003 (two books were purchased), and a human resource to develop the software prototype.

4. DESIGN OF SOFTWARE PROTOTYPE

A software design model was created for the evolutionary software prototype that included a statechart for the WorkRequest object (OOD using UML (Unified Modelling Language) notation), an ERD (Entity-Relationship-Diagram) for relational database design supplemented by associated notes and business rules, data dictionary, and normalisation, a layered sub-system architecture diagram (OOD using UML notation), a sub-system diagram (OOD using UML notation), use-case diagrams (OOD using UML notation), use-case descriptions (OOD), and references to data flow diagrams and specific system requirements specifications using standard forms (used as functional models to represent functional decomposition and to assist modular decomposition).

5. REALIZATION

Implementation

The design was initially tested and implemented as a stand alone single user Microsoft Access desktop database on a single PC (where both the client and server exist in the same PC) and afterwards additional resources were found to implement the same design as a multi-user (two-tier client server system) Microsoft Access desktop database on a Windows network server with multi-user database access from connected Windows client PC’s running Access 2003. The implemented software prototype is a Microsoft Access desktop database that is known as the Work Request Database. The file name that contains the Work Request Database is ‘Work Request Database.mdb′.

The design was initially implemented exactly as per the design. After implementing the design four evolutionary enhancements were made to the design, two of the enhancements were for convenience and two of the enhancements were out of necessity. The enhancements made to the design are described below.

Enhancement for convenience: The attributes WorkRequestJobStatus and WorkRequestDateCompleted from the WorkRequest table were originally planned as only inputs into the function Create/Edit Implementation And Implementation Details, however, it was decided that it would be more convenient to allow the function Create/Edit Implementation And Implementation Details to also output the attributes WorkRequestJobStatus and WorkRequestDateCompleted to the WorkRequest table. To implement this enhancement the form called Create/Edit Implementation And Implementation Details was changed to allow the user to input new values for the WorkRequest table attributes WorkRequestJobStatus and WorkRequestDateCompleted.

Enhancement for convenience: For convenience an additional option was created on the Work Request Database Main Switchboard to close the database and exit the application. Although the database could be closed by using the Windows 'X' close window icon having a single button to close the database and exit the application was a convenient enhancement.

Enhancement out of necessity: In the Create/Edit Implementation And Implementation Details function (implemented as a form called Create/Edit Implementation And Implementation Details) the selection of the WorkRequestNumber to be implemented was enhanced by creating and using an Access 2003 query called Work Request Select Query Without Matching Implementation. The use of the query called Work Request Select Query Without Matching Implementation was a necessary enhancement as this guaranteed that only WorkRequestNumbers that had not been implemented could be selected to be implemented.

Enhancement out of necessity: The original project specification planned to build a software prototype with Access 2003 without manually writing VBA code as there was not believed to be sufficient time in the project to learn both Access 2003 and VBA. At implementation time it was necessary to change this strategy as it was found that Microsoft Access default error trapping was acceptable for simple Microsoft Access forms but was quite poor for more complex forms such as parent forms with embedded child subforms. As the project was ahead of schedule two weeks were spent to learn and apply VBA code to perform essential error trapping on the more complex Access forms in the database.

Testing the Software Prototype Using Test Data

White box testing (also known as structural testing) is a testing method in which tests are derived from knowledge of a software’s structure and implementation, the objective of white box testing is to ensure that each independent program path is tested at least once. An independent program path is a path that traverses at least one new edge in a program flow graph (this is a graph with nodes representing decisions in the program flow and edges showing the flow of control) (Sommerville, 2001, pp.447-450). The program code of a system must be available to be able to perform white box testing. Black box testing (also known as functional testing as the tester is only concerned with the functionality of the software) is a testing method where the tests are derived from the program specification without any knowledge of the implementation of the software being tested, test data is input into the system and the outputs are verified (black box testing is done without knowledge of the program code of a system) (Sommerville, 2001, p.443). As nearly all of the application logic for the Work Request Database was written with Access 2003 most of the application program code was not available to perform white box testing (only a small amount of program code was available, this code was VBA class module code scoped to individual reports or forms), therefore it was decided that white box testing could not sensibly be performed (as most of the independent program paths were hidden inside Access 2003). It was decided that black box testing would be used for the project, all of the black box software tests performed on the Work Request Database software prototype were successful.

6. EVALUATION

Successes and failures of MS-Access 2003 prototype to meet the requirements

To review the successes and failures of the software prototype to meet the system requirements each requirement was reviewed to see if the software prototype had satisfied each requirement. The review of requirements covered both functional requirements and non-functional requirements, to ensure completeness each requirement was traceable by a requirements identifier and associated user requirement identifier.

Traceable requirements satisfied by the software prototype

All of the functional requirements agreed with the project sponsor to be included in the evolutionary software prototype were successfully satisfied. All of the non-functional requirements agreed with the project sponsor to be included in the evolutionary software prototype were successfully satisfied. As shown below by Figure 5 (Appendix A) the evolutionary software prototype achieved a weighted score of 95 out of 100 using the same evaluation criteria as used to evaluate the suitability of COTS solutions, this exceeded the expected weighted score of 90 out of 100.

As shown below by Figure 6 and Figure 7 (Appendix A) the evolutionary software prototype also exceeded the target weighted score for all functional and non-functional requirements by satisfying some of the requirements that were agreed with the project sponsor to be excluded from the evolutionary software prototype.

Traceable requirements not satisfied by the software prototype

All of the traceable requirements that were agreed to be included in the software prototype were successfully included in the software prototype.

Suitability of MS-Access 2003 as a software prototyping tool

All of the requirements that were agreed to be included in the software prototype were successfully included in the software prototype, this demonstrates that it is feasible to use MS-Access 2003 as a software prototyping tool.

Useful features of MS-Access 2003 for developing a software prototype

We found the following features of MS-Access 2003 particularly useful when developing the Work Request Database software prototype:

- Database: Access 2003 can be used to create either a single user or multi-user relational database. If a software prototype is going to have a reasonable level of depth then a database is often required to provide efficient storage and retrieval of the data used by a software prototype.

- Scalability: A software prototype can be designed as a Microsoft Access desktop database for a small number of users (no more than 20 concurrent users is recommended for a Microsoft Access desktop database) and afterwards options are available to upsize a Microsoft Access desktop database to handle more users. Tools such as the Microsoft Access database upsizing wizard can be used to upsize a Microsoft Access desktop database into an alternative two-tier client server solution that uses Microsoft SQL server.
- User Interface: A software prototype developed in Access 2003 with end-user involvement can provide an effective solution for developing a user interface.
- Speed of development: Software prototypes are often able to reduce the cost of getting a software product to the market by reducing the time for the design process, for maximum benefit a software prototype has to be developed as quickly as possible. Access 2003 contains many wizards that help to speed up the development of a software prototype. In the project we used wizards to create Access 2003 forms, reports, queries, and command buttons. Once an Access 2003 object has been created with a wizard it can normally be tailored by editing the object in design view. Although a wizard is also available to create Access 2003 tables we preferred to create each table manually in design view using the data dictionary as a source document. We also used the Access 2003 Switchboard Manager to create a switchboard form (a menu is known as a switchboard in Access 2003).
- Visual development: Access 2003 supports a visual programming approach that allows a software prototype to be built by defining objects (or components) such as switchboards (menus), forms, form controls, and command buttons.
- Object Dependencies: It is normal to expect changes in software systems as part of the systems lifecycle, this is also true for software prototypes (especially evolutionary prototypes). Access 2003 has a feature called object dependencies that identifies which objects are dependent on a specified object, this feature is very useful when analysing the impact of system changes.
- Import/Export: Although not used in the project Access 2003 provides facilities to import and export data in the Microsoft Office environment. Examples of possible sources to import data from into an Access 2003 table include a MS-Excel worksheet, a text file, another Microsoft Access database, a HTML file, and a XML file (Online Training Solutions Inc, 2003, pp.58-73). The ability to import existing data into a new Access 2003 table may be useful to speed up the development of a software prototype.
- Data Validation: Access 2003 has several features that help to keep information accurate including table attribute data type settings, input masks, field validation rules, and lookup lists (sometimes referred to as combo boxes) (Online Training Solutions Inc, 2003, pp.146-166).
- Event driven application logic: Access 2003 allows event driven application logic (written in VBA) to be added to certain object types such as form controls, event driven application logic allows additional application logic to be specified that would not be possible if just Access 2003 was used. An example of where event driven application logic is useful when developing a software prototype is to have the ability to send a tailored message in a message box to the screen after the user has clicked in a particular form control (VBA code could be created for the on-click event for a specified form control, when the user clicks on the specified form control the on-click event is fired and the VBA code behind the on-click event is executed).
- Security: In the project multi-user access to the software prototype was secured by using NTFS security permissions (all members of the IT team were made members of a user group that had update permissions to the software prototype), however Access 2003 also provides some security features that might be useful for securing a software prototype. Security features provided by Access 2003 include encrypting/decrypting a database, assigning a password to a database, creating an Access 2003

workgroup, securing VBA code with a password, and securing a database by making a MDE (Microsoft Database Executable) file. In a MDE file users can not use design view for forms, reports or modules, or change references to other objects or databases, or change VBA code (Online Training Solutions Inc, 2003, pp.226-255).

Potential disadvantages of using MS-Access 2003 for developing a software prototype

In order to give a balanced opinion of Access 2003 as a prototyping tool we believe that it is worth mentioning the features of Access 2003 that caused concern whilst developing the software prototype. Based on the experience gained in the project the following negative features were identified when using Access 2003 as a software prototyping tool:

- Default error trapping: Access 2003 default error trapping is suitable for simple Access 2003 forms but is not suitable for more complex Access 2003 forms such as parent forms with embedded child subforms. We found it necessary to learn and apply some VBA class module code (scoped to within individual Access 2003 forms) to provide satisfactory error trapping and error prevention, without using some VBA code the software prototype would not have been successful.
- VBA class module code method restrictions: The VBA SetFocus method moves the focus to a specified Access 2003 form control. The VBA SetFocus method does not work for certain types of form controls on an Access 2003 form (such as a combo box), this caused frustration when performing error trapping or when trying to automate the sequence in which each form control is accessed.
    - Environment: Care needs to be taken regarding the environment in which an Access 2003 database is developed. During the project we experienced a problem where the code produced by the command button wizard would not operate without errors (2nd author was receiving errors such as 'The command Select Record is not available now' and 'The command Delete Record is not available now'). First of all we executed the VBA code produced by the command button wizard in debug mode, no errors could be found in the VBA code. After debugging the VBA code we looked for any occurrences of the problem on the Microsoft Website, no problem information could be found matching the symptoms we was experiencing. To solve the problem we had to think differently, maybe the problem was an environmental problem as we had been working on more than one PC when developing the software prototype. we re-opened the database in design mode after ensuring that we was opening the database with Access 2003 and not a previous version of Access and deleted and recreated the failing command button with the command button wizard. Exactly the same VBA code was produced as per the VBA code that was previously failing but this time the VBA code worked. we concluded that the problem was an environmental problem that was probably due to opening the software prototype with an earlier version of Access by mistake (causing incorrect DLL's to be used).
- Exclusive design mode use: To avoid losing modifications Microsoft Access 2003 requires exclusive access to a database when using a database in design mode (it is not advisable for more than one software developer to simultaneously open the same Access 2003 database in design mode). This restriction results in Access 2003 being not well suited for development projects where more than one software developer requires simultaneous design mode access to the same Access 2003 database therefore Access 2003 is not well suited for designing software prototypes for larger projects that require more than one software developer to have simultaneous design mode access to the same Access 2003 database.
- The need to use some VBA code: It is not recommended to rely entirely on Access 2003 to build a software prototype. It is possible to create a software prototype using Access

2003 without manually writing any VBA code, whilst this approach may speed up the development of a software prototype the completed software prototype may suffer from the following two problems:

- Based on my experience of developing a software prototype with Access 2003 it is likely that using VBA code will be required to make a software prototype acceptable to the user.
- If only Access 2003 is used to create a software prototype then all the application logic is hidden inside Access 2003, this can be disadvantageous as the developer can not see the application logic, also the programming code is not available for white box testing.

7. CONCLUSION OF THE SUITABILITY OF MS-ACCESS 2003 AS A SOFTWARE PROTOTYPING TOOL

As demonstrated by the successful results of this project Access 2003 can be used as an acceptable tool to build a software prototype. The use of some VBA code in conjunction with Access 2003 allows a more user friendly software prototype to be developed (for example the use of VBA with Access 2003 can provide better error handling logic and more user friendly error messages than by using Access 2003 without any VBA code). In order to achieve a better quality software prototype it is recommended to use Access 2003 in conjunction with some VBA code, this strategy is particularly important if an evolutionary software prototype is being developed that will be retained as the final system. Software prototypes can still be developed with Access 2003 without writing any VBA code however this strategy is better suited to throw-away software prototypes that are not retained as the final system.

8. STRENGTHS AND WEAKNESSES OF THE PROJECT AS CARRIED OUT

The project as carried out was a success, therefore the strengths of the project as carried out are far greater than the weaknesses. The main strengths of the project are as follows:

A software prototype was successfully created (using Access 2003 with some VBA code) and implemented that satisfied all the requirements that were agreed with the project sponsor to be included in the software prototype.

The suitability of Access 2003 as a software prototyping tool was successfully reviewed based on the experience gained through the project.

The completed software prototype provided a successful working software solution that can be used to improve work planning, help desk support, and the recording of software changes in a small IT department.

The project was completed in a timely manner in compliance with the timing plan defined in the project plan and all objectives were achieved.

The main weaknesses of the project are as follows:

Although additional tailored error messages and error handling logic was provided by using VBA code some of the default error messages provided by Access 2003 could have more meaningful text and associated actions that help the user to better understand the necessary corrective actions to resolve an error situation.

Although the project was a success we do not believe that Access 2003 is the best tool to develop a software prototype, we prefer to think of Access 2003 as an acceptable tool to develop a software prototype that is better suited to a single software developer environment. It would be interesting to recreate the Work Request Database software prototype using different software prototyping tools such as the high-level languages Visual Basic (or a more recent version such as Visual Basic.Net) or Java in order to compare how each tool performed as a software prototyping tool in comparison to Access 2003.

9. CONCLUSIONS

Summary of the project as a whole

As described in the project overview the project had two aims, both of these aims were achieved.

The first aim of the project was to

conduct a high level evaluation of a small sample of commercial software solutions as well as to design and implement a bespoke purpose built software prototype built with Microsoft Access 2003 in order to recommend a potential software solution to improve work planning, help desk support, and the recording of software changes in a small IT department. The proposed solution had to satisfy the requirements that were discussed and agreed with the project sponsor in a formal requirements specification document. An evolutionary software prototype was successfully designed and implemented as the recommended solution to improve work planning, help desk support, and the recording of software changes in a small IT department. The evolutionary software prototype did successfully satisfy the requirements that were discussed and agreed with the project sponsor in a formal requirements specification document. The completed software prototype provided a successful working software solution that can be used to improve work planning, help desk support, and the recording of software changes in a small IT department.

The second aim of the project was to review the suitability of Microsoft Access 2003 as a prototyping tool based on the experience gained during the project through building a Microsoft Access 2003 software prototype to meet the requirements agreed with the project sponsor. The suitability of using Microsoft Access 2003 as a prototyping tool was successfully reviewed based on the experience gained during the project through building a Microsoft Access 2003 software prototype to meet the requirements agreed with the project sponsor.

Possible applications and extensions of the work

The following applications and extensions of the work are worth consideration:

- An archiving function could be added to the software prototype to archive records over a specified age.
- It would be interesting to investigate if a front end could be built to allow user work requests to be received by email and automatically integrated into the Work Request Database as open jobs.
- A more detailed understanding of the required search facility could be developed by researching intelligent search techniques and knowledge based systems in order to recommend and build a solution to expand the evolutionary software prototype to include a more intelligent search facility.
- It would be interesting to recreate the Work Request Database software prototype using different software prototyping tools (such as the high-level languages Visual Basic and Java) in order to compare how each tool performed as a software prototyping tool in comparison to Access 2003.

Bibliography

Coronel,C. Rob,P. (2002) Database Systems. 5th ed. USA: Course Technology. ISBN061906269X

Cniche. Request Tracker help desk database system. [Internet]. Available from <http://www.cniche.com/request/index.htm> [Accessed 21 May 2005a]

Cniche. Running Request Tracker on a Network. [Internet]. Available from < http://www.cniche.com/request/network.htm> [Accessed 21 May 2005b]

Dawson, C.W (2000) The Essence Of Computing Projects A Students Guide. Norfolk: Pearson Education Limited. ISBN013021972X

Dictionary.com. Data Model. [Internet]. Available from:
< http://dictionary.reference.com/search?q=data+model&r=67> [Accessed 6th April 2005]

Gonzalez L. & Giachetti R. & Ramirez G. (2005) Knowledge management-centric help desk: specification and performance evaluation. Decision Support Systems [Internet], no. 40, 2, pp 389-405. Available from: <http://www.sciencedirect.com/science/article/B6V8S-4CHJ8R2-1/2/d701f9f268a422eacd82ed0190432006> [Accessed 30 May 2005]

HMSO. Data Protection Act 1998. [Internet]. Available from:

<http://www.hmso.gov.uk/acts/acts1998/80029--a.htm#1> [Accessed 6 May 2005]

Information Commissioner. Data Protection Act Factsheet. [Internet]. Available from: <http://www.informationcommissioner.gov.uk/cms/DocumentUploads/Data%20Protection%20Act%20Fact%20V2.pdf> [Accessed 6 May 2005a]

Information Commissioner. Notification Exemptions, A Self Assessment Guide. [Internet]. Available from: <http://www.informationcommissioner.gov.uk/cms/DocumentUploads/Notification%20Exemptions%20-%20A%20Self%20Assessment%20Guide.pdf> [Accessed 6 May 2005b]

Information Commissioner. Freedom Of Information Factsheet. [Internet]. Available from: < http://www.informationcommissioner.gov.uk/cms/DocumentUploads/FOI%20factsheet.pdf> [Accessed 9 March 2005c]

Information Commissioner. About The Freedom Of Information Act. Available from: <http://www.informationcommissioner.gov.uk/eventual.aspx?id=74
> [Accessed 9 March 2005d]

Information Commissioner. Data Protection. Available from: < http://www.informationcommissioner.gov.uk/eventual.aspx?id=34> [Accessed 9 March 2005e]

Information Commissioner. Information Commissioners Office. Available from: <http://www.informationcommissioner.gov.uk/eventual.aspx?id=6786&expmovie=1> [Accessed 9 March 2005f]

Information Commissioner. What the Regulations Cover. [Internet]. Available from: < http://www.informationcommissioner.gov.uk/eventual.aspx?id=94> [Accessed 9 March 2005g]

Intuit. Intuit Information Technology Solutions: Track-It! 6.5. [Internet]. Available from <http://www.itsolutions.intuit.com/Track-It.asp> [Accessed 21 May 2005]

Lederer A. & Prasad J. (1995) Causes of inaccurate software development cost estimates. Journal of Systems and Software [Internet], no.31, 2, pp. 125-134. Available from: <http://www.sciencedirect.com/science/article/B6V0N-404RP1C-C/2/e878da8a26708782edabe6aaf7744be2> [Accessed 8 May 2005]

Limoncelli, T. (1999) Deconstructing User Requests And The Nine Step Model. In: LISA '99: 13th Systems Administration Conference, 7-12 November 1999, Washington, USA. Washington. The Usenix Association. pp.35-44.

Marcella R. & Middleton I. (1996) The Role Of The Help Desk In The Strategic Management Of Information Systems. OCLC Systems & Services [Internet], no. 12, pp. 4-19. Available from: <http://miranda.emeraldinsight.com/vl=2576611/cl=16/nw=1/fm=docpdf/rpsv/cw/mcb/1065075x/v12n4/s1/p4> [Accessed 17 May 2005]

McGregor J. (2004) Software Architecture. Journal Of Object Technology [Internet], no. 3, 5, pp 65-77. Available from: <http://www.jot.fm/issues/issue_2004_05/column7> [Accessed 30 May 2005]

Microsoft Help And Support. How to keep a Jet 4.0 database in top working condition. [Internet]. Available from: <http://support.microsoft.com/default.aspx?scid=kb;en-us;303528> [Accessed 8 March 2005a]

Microsoft Help and Support. How to obtain the latest service pack for the Microsoft Jet 4.0 Database Engine. [Internet]. Available from: <http://support.microsoft.com/default.aspx?scid=kb;en-us;239114> [Accessed 2 September 2005b]

Appendix A

Req. ID	Require-ment Type	Priority H=High M=Med L=Low	Requirement Description	Associated User Require-ment ID	In-clude In Soft-ware Proto type Y/N
FR0 01	Functional	H	The software product should allow users to create, update and delete user work requests.	UR002	Y
FR0 02	Functional	M	The software product should validate critical data entered by users by applying suitable attribute validation rules where validation can be sensibly applied.	UR003	Y
FR0 03	Functional	H	The software product should prevent data redundancy or multiple occurrences of the same data. This functional requirement must be satisfied by using a normalised relational database structure.	UR004	Y
FR0 04	Functional	H	The software product should provide a means to update the most current status for a user work request. The status of a user work request can be open, rejected, or closed.	UR008	Y
FR0 05	Functional	H	The product should be able to report open jobs (open user work requests) sorted by the person (computer specialist) that each job (user work request) is allocated to.	UR009	Y
FR0 06	Functional	H	The product should be able to report completed jobs (closed user work requests) sorted by the work request site, work request department, work request user (the employee who requested the work), and work request completion date within a date range that is specified when the report is run.	UR010	Y
FR0 07	Functional	H	The product should record user work requests.	UR0011	Y

FR0 08	Functional	H	The product should record (when required) implementation details to satisfy software change control requirements.	UR0012	Y
FR0 09	Functional	L	Subject to the time constraints imposed on the project it would be useful for the product to have a search facility to search previous user work requests to see if a particular problem has occurred before. A more detailed understanding of the required search facility is required before this requirement can be satisfied, therefore the evolutionary software prototype is not expected to include this requirement.	UR0014	N

Table 1: Summary of functional requirements

Table 2: Summary of non-functional requirements

Req. ID	Non-functional Requirement Type	Priority H=High M=Med . L=Low	Requirement Description	Associated User Requirement ID	Include In Software Prototype Y/N
NFR001	Non-functional Product Usability Requirement	H	The software product should be designed to be suitable for members of a small IT department that consists of 5 users or less.	UR001	Y
NFR002	Non-functional Product Usability Requirement	H	The software product must have the capability of being used concurrently by a maximum of 5 users through shared network access. The software prototype was not expected to be able to demonstrate this capability as it was planned to be built as a stand alone solution due to physical resource constraints.	UR005	N
NFR003	Non-functional Product Usability Requirement	L	Ideally the software product should have the future possibility of being expanded to handle more than 5 users. The software prototype will not include this requirement.	UR007	N

NFR004	Non-functional Product Usability Requirement	H	The system must be responsive; a typical user work request must be able to be recorded in the database in less than 15 minutes.	UR0013	Y
NFR005	Non-functional Product Reliability Requirement	M	The system must be reliable; ideally system downtime should be less than 1% per year. The evolutionary software prototype should be able to demonstrate reliability over short periods of use; however, there is not sufficient time within the project to monitor system downtime over the duration of a year.	This requirement has been included as it is in the interest of good practice.	N
NFR006	Non-functional Organisational Requirement	H	Due to budget restrictions the software prototype must be built using MS-Access 2003 on a stand alone Windows based PC.	UR006	Y
NFR007	Non-functional External Legislative Requirement	H	Brief considerations should be made for any legal requirements that may be relevant to the project (such as the UK Data Protection Act).	General constraint	Y
NFR008	Non-functional External Ethical Requirement	H	The system must be ethically acceptable to the users.	This requirement has been included as it is in the interest of good practice.	Y

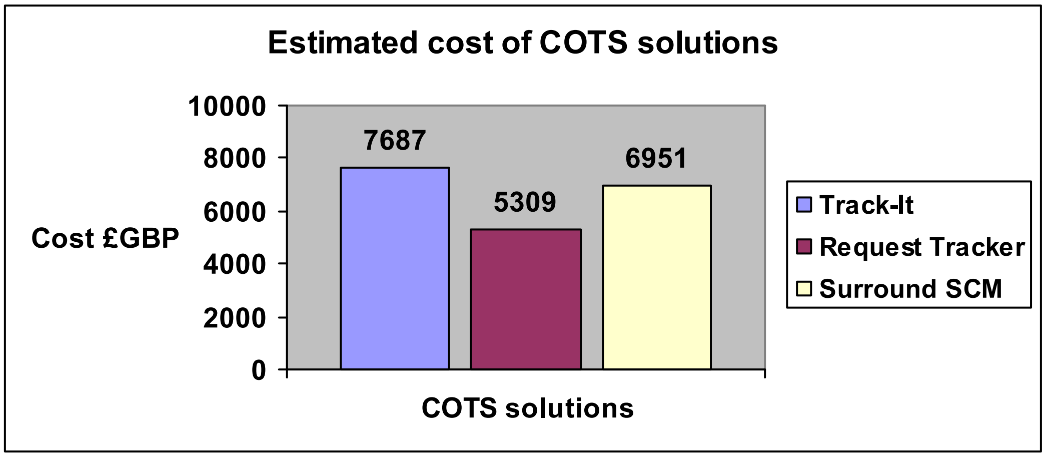

Figure 1: Estimated cost of COTS solutions

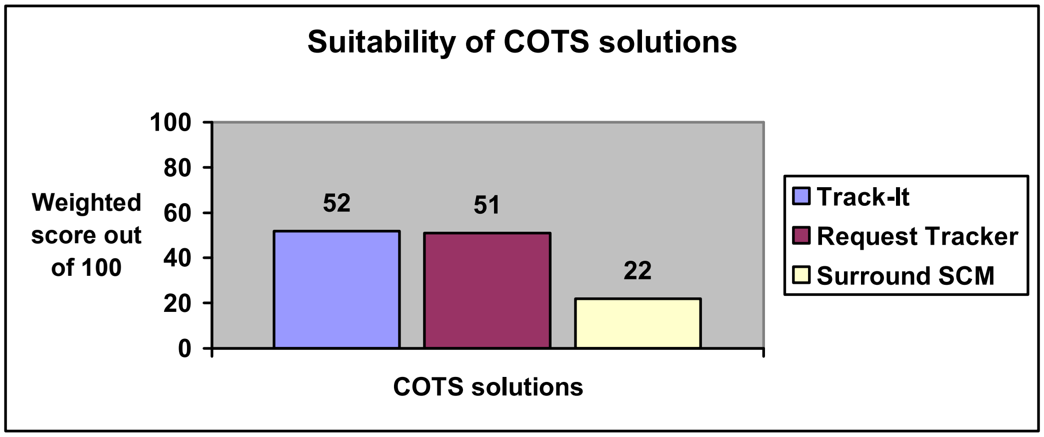

Figure 2: Suitability of COTS solutions

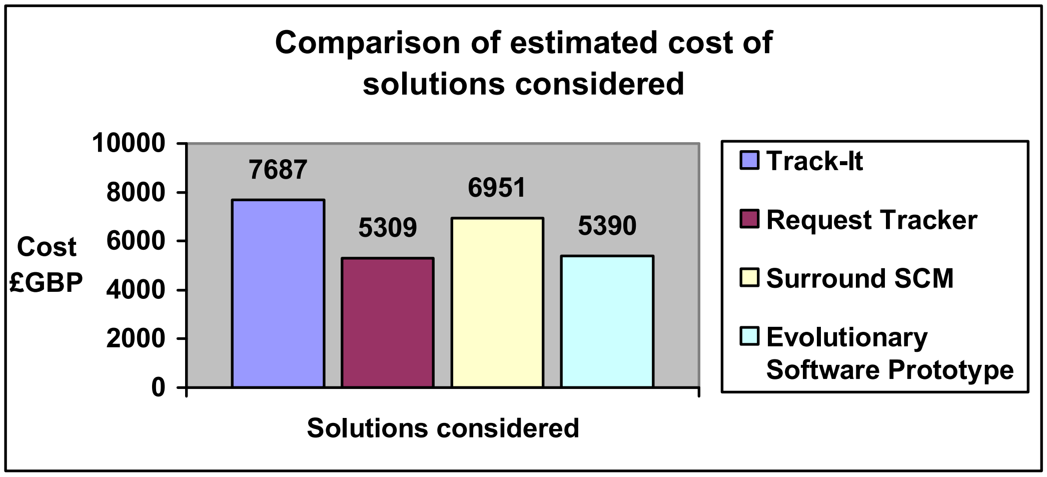

Figure 3: Comparison of estimated cost of solutions considered

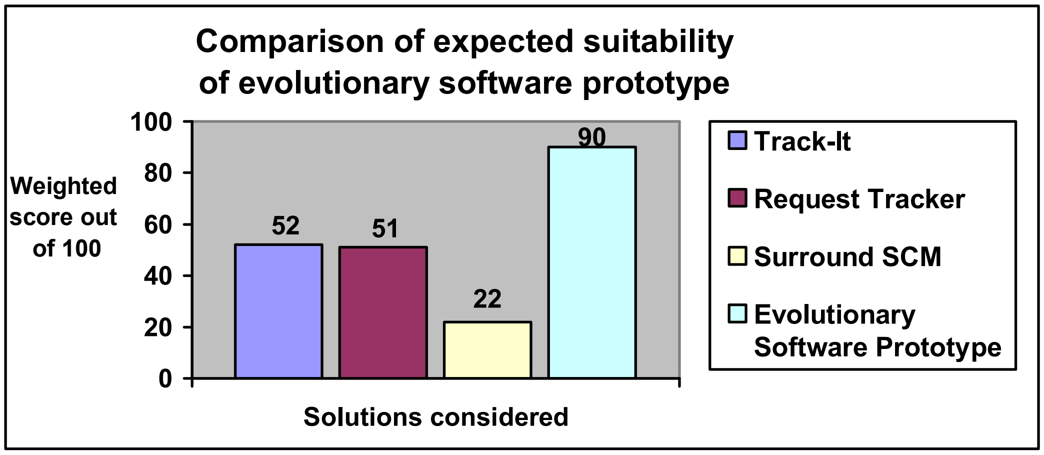

Figure 4: Comparison of expected suitability of evolutionary software prototype

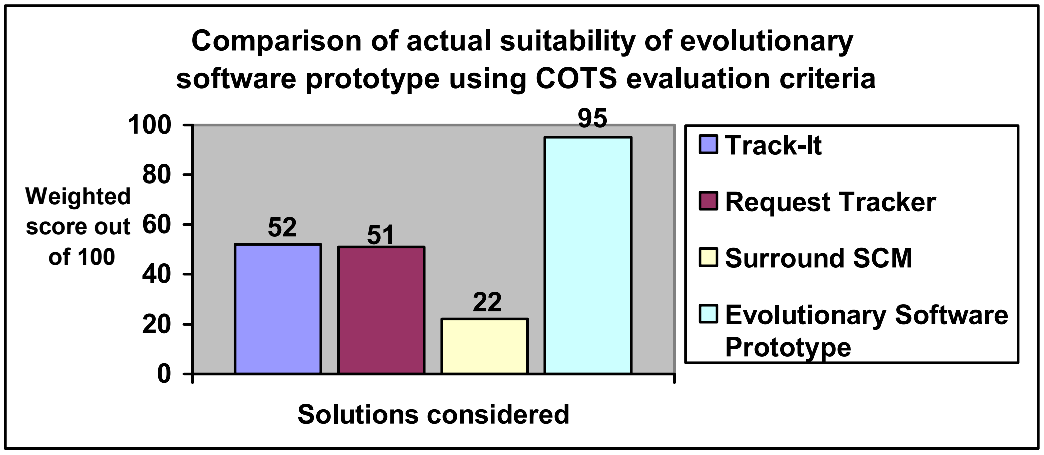

Figure 5: Comparison of actual suitability of evolutionary software prototype using COTS evaluation criteria

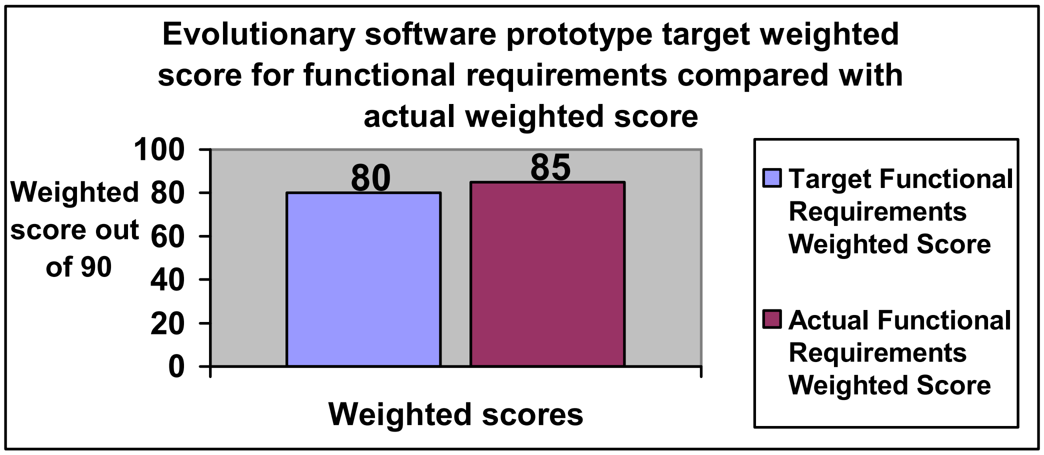

Figure 6: Evolutionary software prototype target weighted score for functional requirements compared with actual weighted score

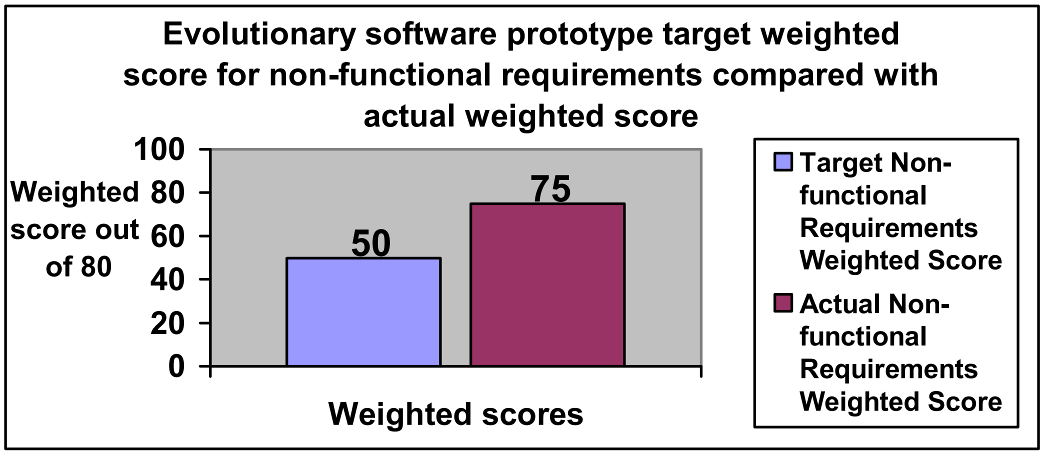

Figure 7: Evolutionary software prototype target weighted score for non-functional requirements compared with actual weighted score