Tracing Requirements Errors to Problems in the Requirements Engineering Process[1]

A.G. Sutcliffe, A. Economou* and P. Markis*

Centre for HCI Design
City University
Northampton Square
London EC1V 0HB
United Kingdom
Tel: +44-171-477 8411
Fax: +44-171-477 8859
e-mail: a.g.sutcliffe@city.ac.uk

*EDP Center for Social Insurance
101 Syngrou Av.
GR-117 45 Athens
Greece

Abstract

A case study of requirements engineering practice is reported. The application, a decision support system for the Greek Ministry of Health, was investigated by studying the process of requirements analysis through to design and implementation. A usability analysis was then conducted on the designed system with the users. Several usability problems were discovered, and interviews uncovered further problems with the system that could be attributed to failure in requirements engineering. Even though requirements were explicitly stated and the system was an evolution from an existing legacy system, functionality was defective and usability was poor. The client's prime concern for redeveloping the system was to improve usability; unfortunately communications problems in the RE process meant that the developers did not appreciate this. The implications for RE methods and understanding the RE process are discussed.

Keywords: case study, requirements engineering, problem analysis, usability.

1. Introduction

There is considerable evidence of requirements engineering failure in projects that are delivered late, overbudget and do not meet user requirements, for instance London Ambulance Service [1]; however, few studies have investigated the link between requirements quality and the requirements engineering (RE) process. The London Ambulance Service failure highlights high level problem of poor communication, insensitive management and lack of user participation, but such disaster stories may be atypical. If we are to improve RE practice the causation of poor requirements capture and validation has to be traced to weak links in the process in a wider variety of systems. An early study of software engineering practice, conducted by Curtis et al. [2], found that accurate problem domain knowledge is critical to the success of a project and requirements volatility causes major difficulties during development. Lubars et al. [3] reviewed the state of RE practice in industry, concluding that organisational solutions to RE problems are preferred over technology, and that

[1] This research has been partly funded by the European Commission ESPRIT 21903 'CREWS' (Co-operative Requirements Engineering With Scenarios) long-term research project.

ambiguous and changing requirements pose significant problems. Waltz et al. [4] studied a single project that developed an object server for a programming environment. The team's interactions over a four month period were dominated by communication and knowledge acquisition problems. Team members forgot issues they had discussed, made different assumptions, while meetings lacked facilitation and direction. The requirements process 'shut down' when the project ran out of time.

The need for effective requirements engineering has also been noted the development of Clinical Decision Support Systems (DSS) by Timpka and Johansson [5] who interviewed medical and system development professionals from different countries. Their main findings were that use of problem-oriented methods was seen as essential, but that employing these methods was hindered by constraints in project budgets. The study also showed a consensus on the importance of 'end-user' involvement, i.e. that medical practitioners should be able to strongly influence requirements analyses.

El Emam and Madhavji [6] in a field study of requirements engineering concluded that both technical and non-technical issues are important. In particular, they point to problems in the lack of trained personnel, inadequate methods to support the planning stage of RE, and difficulty in securing an adequate level of user participation. Al-Rawas and Easterbrook [7] investigated communication problems in RE, and provide evidence that organisational and social issues influence the effectiveness of communication and therefore impact on the overall success or failure of the RE process. One way communication, no support for informal communication, inappropriate notations and methods, and organisational barriers were some of the causes for communication breakdowns.

Overall, these studies point to a number of problems, but in particular poor communication, lack of a systematic approach, need for domain knowledge, and changing requirements are highlighted as the main problems faced during requirements engineering. Although previous studies have investigated problems in the RE process, we have few insights into how failures in the RE process impact on product quality and system acceptance, apart from well known, and possibly atypical disaster stories [1]; furthermore, no studies have considered requirements in the perspective of overall product quality, e.g. usability and utility, nor have they attempted to link poor product quality to process failings. Whether a system is acceptable for users or not is a function not only of utility but also of usability [8], so investigation of product acceptability needs to account for its functionality and ease of use. To diagnose the reasons for system failure a combination of both requirements engineering and usability analysis is required.

This paper reports an ecological study of the requirements engineering process that tries to account for why a system was unsatisfactory in both functional and operational terms. It also attempts to identify reasons for those failures in the design, and the requirements engineering process. The paper is structured as follows. Section 2 introduces the application that was studied; section 3 describes the analytic methods employed; this leads into section 4 that reports the usability evaluation and assessment of the requirements analysis for the case study product, a health sector executive information system. Section 5 then traces the RE process to discover the reasons for failure and their impact on product quality.

The paper is forensic in nature. We start with an existing system and gather evidence about its operational and functional acceptability. From this evidence we go back in time to investigate the causes and motivations that underlie observed defects in an implemented system. First, we introduce the case study.

2. Case Study Application HIPPOCRATES

HIPPOCRATES is a Windows-based Executive Information System that collects, distills and presents critical information on hospitals, doctors and patients throughout Greece to support the decision making process of health planners and executives of the Ministry of Health.

The primary use of the system is to assist in health planning and monitoring.

The system stakeholders were:

- Six Health planners of the Greek Ministry of Health and Welfare who are the primary end-users.

- About 25 Departmental managers and administrative staff who use the system for making specific inquiries such as hospital staff vacancies, patients flow, bed occupancies, and expenses on specific subjects. The managers usually use the system through intermediaries.

- Private enterprises (pharmaceutical, medical equipment sellers), secondary users of the system who information held in the HIPPOCRATES database to satisfy the needs of their marketing departments. They are usually interested in consumption in medicine, equipment, etc.

- Other bodies outside the Ministry, e.g. local and international independent consultants, health care researchers taking part in EU projects, college staff or students doing research on health, international health bodies.

- The Ministry's Central Health Council (CHC) who are responsible for providing information on health matters to meet the needs of the primary and secondary end-users.

- Health Informatics Group (HIG) is the Informatics unit of the Central Health Council, consisting of the manager, three analysts/programmers and two data entry clerks. The HIG analysts conducted the initial analysis and design for HIPPOCRATES and implemented the older version in Cobol. They are responsible for the introduction and maintenance of the system in the Ministry and in external installations.

  HIPPOCRATES was financed by the Ministry of Health through its Central Health Council (CHC). and this study investigated an upgrade of the legacy Cobol system that had been developed by the CHC's Health Informatics Group (HIG). The initial analysis for this version was performed by a team from the Health Informatics Group, consisting of the HIG manager and two developers.

The software contractor commissioned to build the product was DBSoft S.A. DBSoft's staff had no previous experience with Health EIS. The development team consisted of the project manager/senior developer and three other developers. DBSoft followed the standard IBM methodology, Solution 2000. The two parties signed a fixed price, pre-paid contract. It was a two-phased contract divided into an analysis phase and implementation phase.

The legacy system was composed of two sub-systems: the Data Entry part that allows HIG personnel to enter and edit data and codings, and the Information retrieval (IR) sub system. The IR sub systems contained health data grouped into four homogeneous subsystems; concerning functional information (general statistics on ward resources and their utilization), personnel data, financial information (revenue and expenses by ledger category), and patient data (personal and disease related data for each hospitalised patient). The structure of the new system was based on the legacy system. Further details of the system can be found in Markis [17].

3. Analysis Methods

This section examines the research methods employed in the project. First, an overview of the methods is given and then each method is dealt with in turn. These are:

- Diagnostic testing of the human-computer interface.
- Questionnaires.
- Interviews.
- Documentation analysis.

3.1 Diagnostic evaluation

To reveal the usability problems of the design we used the diagnostic evaluation method, Model Mismatch Analysis [9], because it helps to identify the design features responsible for the problems. Furthermore, the method recognises errors in design functions thereby giving a link through to failure in requirements analysis.

The evaluation sessions took place in the natural work setting of the users. Five novice users participated in the 90 minute sessions. They had only used PCs for simple tasks such as word processing or spreadsheet manipulation. Only 40% of the users had a working knowledge of Windows and all had little experience with Executive Information Systems. They had already been using the system for two weeks, so no training was necessary. The evaluation tasks, specified in co-operation with the users, were as follows:

Task 1. Produce a summary report on the status of selected hospitals. The data in the report shall be a combination of hospital general characteristics, financial data and health indicators. Then drill-down to more detailed information per hospital.

Task 2. Produce a summary report containing all doctors of a specific specialty in a selected geographical area.

Task 3. Produce tables of pre-defined format of a certain data category for selected hospitals.

The users were observed while they carried out the tasks and any usability problems noted, following the critical incidents and breakdowns classification of Monk and Wright [10]. Further details of the evaluation session design can be found in [11]. Mini-interviews were held after the test sessions to follow up on reasons for errors.

The errors and problems observed were analysed using the error taxonomy proposed by Sutcliffe [11], as follows:

- Misleading cue: prompt or cue gives the user ambiguous or incorrect information.
- Poor/absent feedback: change in the system's state after an action is not clear; poor predictivity for next action.
- Hidden functionality: command exists, but the user cannot find it.
- Missing functionality: command for the user's action is not in the system.
- Inappropriate functionality: command exists, but it does not do what the user wants.

The latter 3 categories are all attributable to failures in the RE process, while problems in the first 2 categories should be largely eliminated by user testing and iterative prototyping, also a recommended RE practice. The design features responsible for each error were noted and error frequencies for each user by design feature were totalled. The most important problem cases are when a design feature has a high error total for all users [11].

3.2 Interviews

Semi-structured interviews were carried out with 8 individuals drawn from the end-users, the Ministry's IT personnel and the software house development team (suppliers). To ensure a representative sample, both project managers and technical personnel were interviewed. The personnel interviewed included the HIG manager and two of the analysts, two advisors and an administrative assistant to the Minister's Office, the Project Manager and Account Manager of DB Soft SA. Users responded to a set of questions covering their main tasks, the nature of their work and the decisions they needed to take. They also provided information on their role in the process of specifying and designing the system and their view of the requirements.

3.3 Document analysis

The documents listed in Figure 1 produced by the requirements engineering process were inspected. The information gathered from the documentation analysis was used to provide lists of functional and non-functional requirements which were checked against the results of the usability evaluation to indicate problems in the requirements analysis stage. The same documents were given by the requirements engineers to the

developers (DBSoft). Requirements explicitly stated in the requirement specification were analysed so we could determine explicitly stated and implicit requirements.

4. Results

The results are split into an audit of the system functionality and usability problem analysis. The functionality analysis was compiled from a retrospective analysis of documents, including requirements specification, and interviews with users. The functional requirements were explicitly stated either in the legacy system and its documentation or in the tender document for new requirements. Two views of the requirements and the implementation are represented:

- a gold standard of all explicit and implicit requirements that should have been implemented
- requirements that were explicitly stated and system functions that were implemented.

The data model of the system
The description of the processes of the system
The requirements specification documents produced by the development team
The user's guide produced by the development team
The full set of documents of previous system (system description, user's guide, sets of models)
The annual book of health
The contract signed between the Ministry and the Software house
Post-mortem reviews of the pre-existing system

Figure 1 List of documents produced by the requirements engineers.

4.1 High level requirements and the implementation

The high level requirements were to support decisions in health resource management and to provide information retrieval for ad hoc needs. High level requirements describing the necessary support for the users' tasks are rarely made explicit in many requirements documents and the HIPPOCRATES specification was no exception. This section's results were derived from interviews and document analysis to create a requirements critique based on users' comments.

Hospital funding

The financial status of hospitals is monitored and detailed data on revenue/expenses is consulted. Special indicators such as self-sufficiency are computed so that executives can assess the extent to which hospitals are self-sufficient and compare the performance of equivalent hospitals.

Implementation The financial data sub-system contained detailed data on revenue/expenses for every hospital, and the indicator creation facility permits

calculation of all required indicators. Therefore, all the information required was available, and this requirement was satisfied.

Pricing policy

The health planners have to make decisions about pricing policy and investigate alternatives to the current charges, so that the increase in hospital charges is proportionate to the average costs.

Implementation. The financial data sub-system holds information on hospital expenses and calculates the average daily cost of hospitalisation, so this requirement was satisfied.

Man-power planning

Decisions on the staffing of hospitals should take into account both hospital needs and international standards. Information on the man-power of hospitals (doctors, administrators, nurses, other medical specialisations) and indicators such as the average number of beds per doctor, nurses per doctor, percentage of doctors, out of the total number of staff, etc. are required for monitoring and evaluation purposes.

Implementation. HIPPOCRATES holds detailed information on the man-power of hospitals, as well as information on all the doctors in the country. The indicator facility permits the computation of all relevant statistics, such as the average number of beds per doctor. This requirement is satisfied.

Allocation of resources

The allocation of beds and medical equipment to hospitals is decided by health executives who need to have a picture of the number of beds, the occupancy ratio and the number of laboratory examinations per department in each hospital.

Implementation. All the relevant information and statistics are provided in the functional data sub-system.

Acquisition and maintenance of hospital equipment

Alternative policies of decentralised acquisition and maintenance procedures versus centralised ones have to be evaluated.

Implementation. The system does not support this task. No information on equipment suppliers is kept.

Medicine administration policy

To plan administration, the consumption of medicine is monitored and compared between similar hospitals.

Implementation Consumption is monitored in the financial data sub-system, so this requirement is supported.

Hospital expansion

Decisions are made concerning giving out permits for hospital expansion. The bed occupancy ratio is one factor for these decisions.

Implementation. Fully covered in the functional data sub-system.

Policy towards the private sector

The political decision to expand or contain the private sector requires monitoring the status of private hospitals. To support both investors' and the Ministry's information needs, queries are necessary to retrieve the needs of the people by region, the extent to which current public and private hospitals meet these needs and patient data.

Implementation. HIPPOCRATES partially supports this decision-making task. It holds operational, personnel and patient data on private hospitals, but no financial information.

Ad hoc queries

There are a number of ad hoc decisions and information needs which require query facilities and the ability to compute unusual indicators, e.g. turnover interval, etc.

Implementation. A query facility is provided and the system caters for less typical decisions by permitting the executives to compute indicators.

So far it appears that the new system met its requirements apart from some omissions on financial data and equipment suppliers. We now turn to the usability assessment.

4.2 Usability evaluation

The user interface design employed advanced graphics and direct manipulation, as illustrated in figure 2(a) which shows the map interface for selecting geographic areas for information retrieval. This section reports the most common types of errors made by the subjects in the evaluation sessions and the reasons behind them.

4.2.1. Inappropriate functionality

E1. In the health institution selection screen (see Figure 2b) the institution drop-down list box does not provide the facility for entering the institution name. A specific institution can only be selected by browsing through the whole list of institutions. This was time consuming.

E2. The users wanted an indicator to be displayed in a configurable report, but its constituent data elements were displayed as well, thereby cluttering the screen with redundant data.

E3.When detailed data is displayed as part of a configurable report the user has to 'drag' the columns to fit the data. This was confusing for the users.

E4.When a window is minimised not all the application is minimised. As a result the user has to minimise several screens if he/she needs to switch to another application. Moreover, screens are not automatically maximised.

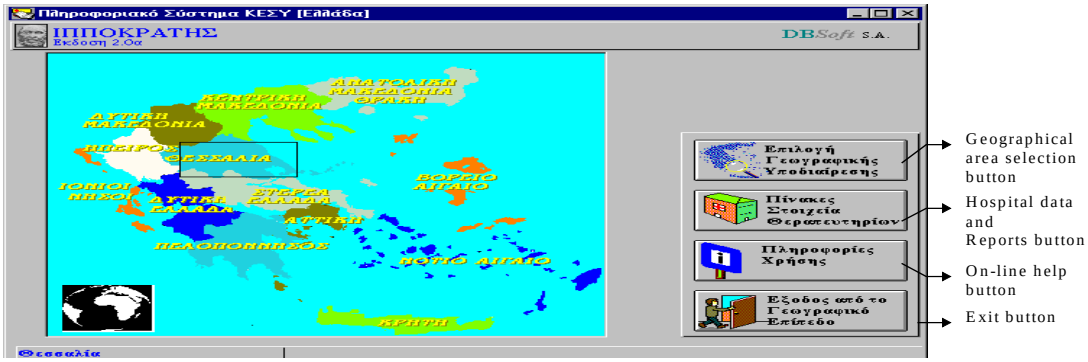

Figure 2(a) Hippocrates User Interface- Map for geographical area specification in data retrieval.

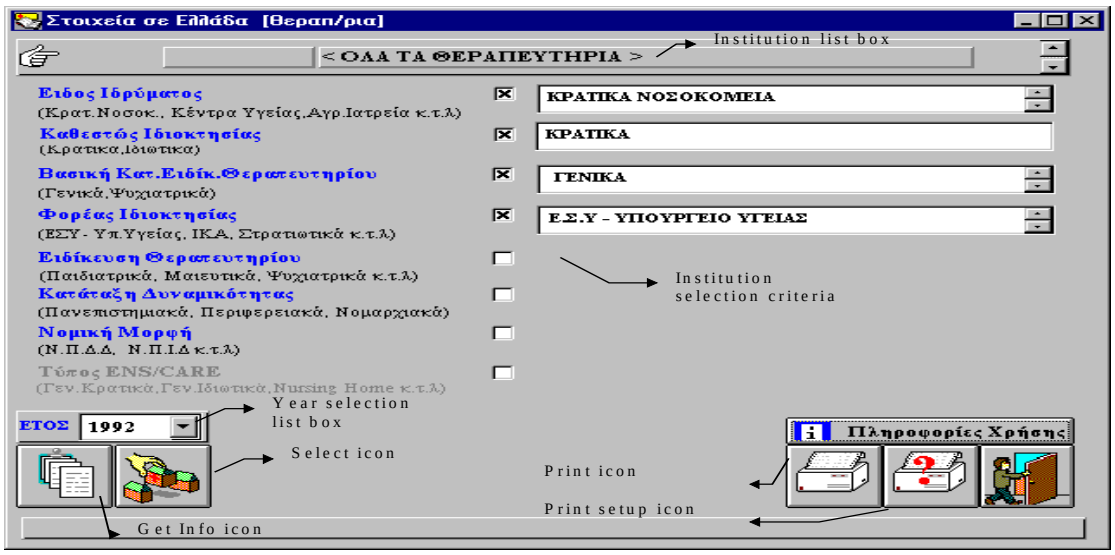

Figure 2(b) Query screen for selecting the type of institution within a specified geographical area.

4.2.2 Hidden Functionality

E5.In the health institution selection screen the options for the different types of institution are not visible to the user. The user has to click the up or down arrow to see the next option.

These problems were all annoying for the users and could have been cured by better usability testing and improved user participation in the design process. They are not, however, direct failings in requirements acquisition, although validation could have eliminated them.

4.2.3 Missing functionality

E6.The financial sub-system could not retrieve certain groupings of data and did not provide summary data grouped under hospital sectors.

E7.In the personnel sub-system, certain information groupings, i.e. all pathology doctors irrespective of staff rank, were not accessible (see Task 2, Step 5).

E8.The search function (drill-down facility) (see Task 1, Step 9) did not allow for retrieval with conditions.

E9.In the screen where health indicators are created users are not allowed to delete or deselect an indicator or edit the calculation after it has been entered, because there is no 'undo' facility.

E10.After a search had been performed using the drill-down facility, no totals were contained in the reports, contrary to the users' request.

E11.Some users complained that they could not drill down to clinic-related information

All these problem are attributable to failures in RE, even though no one problem represented a fatal error. However, the cumulative effect of these missing functions impaired effective system use.

4.2.4 Poor/absent feedback; misleading cues

There were 11 errors in the last two categories. These were all typical HCI usability problems which hinder the ease of use and, inter alia, the effectiveness of the system. Correction and prevention of these errors requires application of user interface design methods and standards (ISO 9241) and prototyping cycles with usability evaluation, but these problems can not be directly attributed to failure in requirements engineering. However, usability is an important non-functional requirement.

The errors encountered during the evaluation sessions classified under usability error type and design feature are summarised in Figure 3, and described in full in the Appendix.

4.3 Requirements and their implementation

The documentation analysis examined the following material:

- The requirements specifications produced by the development team
- The user's guide produced by the development team
- The full set of documents of the pre-existing system (system description, user's guide, sets of models)
- The contract signed between the Ministry and the Software House

Design Feature	Misleading cue	Poor / Absent feedback	Hidden Functionality	Missing Functionality	Inappropriate Functionality
Specific institution selection box					√
Type of institution selection boxes					√
Select icon	√	√			
Get Info icon	√				
Grouping of functional data elements				√	
Grouping of personnel data elements				√	
Drill- down facility				√	
Indicator editing				√	
Criteria display		√			
Indicator display					√
Enter icon	√				
Up & Down Arrow icons	√				
Data elements display	√				
Configurable reports				√	√
Screen navigation					√
Print icons	√				
Selectable areas	√				
Hourglass	√				
Magnifying glass icon	√				
Go Back and Exit icons	√				
Indicator selection check boxes	√				

Figure 3 Analysis of HIPPOCRATES errors by design feature and error type.

From the study of the above material provided by the Ministry's Health Informatics Group (HIG), combined with information from structured interviews, lists of requirements were compiled. These record requirements that had been captured, either as a retained functionality of the old system or as a new requirement. The extent of the implementation of the requirements is summarised in Figure 4.

4.3.1 Data/information requirements

Health data was required for each hospital classified by region, medical specialty, medical service and ranking and type of financial information, including: General statistics, Operational data per hospital sector and ward, Personnel data, Financial data, general ledger categories and Patient data.

Most information requirements were explicitly specified and satisfied but both financial and personnel data gave problems. The system did not provide a facility to retrieve summary financial data grouped under hospital sectors. In the personnel sub-system, users could not access some important information, e.g. pathology doctors. They could not retrieve information on the staff of hospitals grouped under medical specialties. These errors appear in the usability evaluation as Nos 6 and 7.

4.3.2 Functional requirements

Functional requirements were explicitly stated for search output but the functionality for query formulation dialogues was not. The following complete list is:

- To permit data-entry, quickly, reliably, cost-effectively and data loading facilities to enter history data from other databases.
- To retrieve the information for specific institutions or for groups of institutions based on criteria (geographical type).
- To provide extraction functions to export data for manipulation outside the system, e.g. into local spreadsheets.
- To allow the user to specify parameters for IR requests as preformed queries, e.g. year, area level, area unit, and type of health institution and groupings of operational and personnel data by sector/specialty.
- The system should calculate performance indicators covering areas of interest of the health planners, to support decision making.
- To retrieve aggregated data at a high level and lower level detail with a 'drill-down' facility to hospital level, with conditional searches on the same.
- To present retrieved data in maps, charts, graphs and tables formats.
- To produce user-defined, configurable reports and charts to meet special and unpredictable information needs.

	Requs not implemented	Requs spec'd poor design	Requirements declined	Emerging requs
General statistics	●		(d)	
Operation Hosp data			(c)	
Personnel data	(7)			
Financial data	(6) ●		(b)	
Patient data				
Data entry				
Data retrieval		(8) (1) (2)		
Data export/ extraction				
Preformed queries				
Calculate indicators		(3) (9)		
Drill down search		(8) (10)	(a)	(11)
Results presentation				
Configurable reports				
Predefined reports				

(1) numbers refer to usability errors, (a) letters to requirements declined during negoitations

● task support requirements problems

Figure 4 Major requirements and their implementation history with problems detected by usability analysis.

- To produce tables of pre-defined format which contain routinely-demanded data and statistics.

  Several requirements were not adequately implemented. The drill down facility suffered from usability Error 8 as it did not allow conditional searches, and Error 10 because no aggregate totals were contained in the reports. The Calculate indicators facility caused Error 9 in which users could not delete or edit an indicator arithmetic expression because there was no 'undo' facility, and Error 3 in the configurable report: when users only wanted an indicator to be displayed, whereas its constituent data elements were displayed as well. Errors 1 and 3 impaired query formulation and preparation of configurable reports.

4.3.3 Non-functional requirements

The non-functional requirements were not formally captured or documented. The following non-functional requirements were reconstructed from semi-structured interviews with the HIG manager and the project manager of DBSoft, and their implementation status is shown in Figure 5:

- Performance

  Fast response times were required for interactive data retrieval but these were not achieved for some time after implementation when optimisation fixes were designed.

- Portability

  The system should be able to work both in networked PC installations in the Ministry, and stand-alone machines for other users and portable computers for health researchers and consultants.

- Platform

  The RDBMS selected should work in both a client-server environment and stand-alone. It should also be compatible with the Novell 3.11 and Ethernet TCP/IP LANs already installed in the HIG. Software development should use Object Oriented (OO) programming tools and develop for a Windows User Interface environments.

- Security

  The security of the data should be preserved by means of user passwords and users should have access only to the EIS. The data-entry component of the system shall be available only to the HIG staff.

- Accuracy of information

  The accuracy of the information is extremely important as monitoring and planning of health actions should be based on accurate data. Accuracy was partially compromised by deficiencies in data retrieval, poor facilities for data loading and missing information requirements.

- Legacy systems constraints

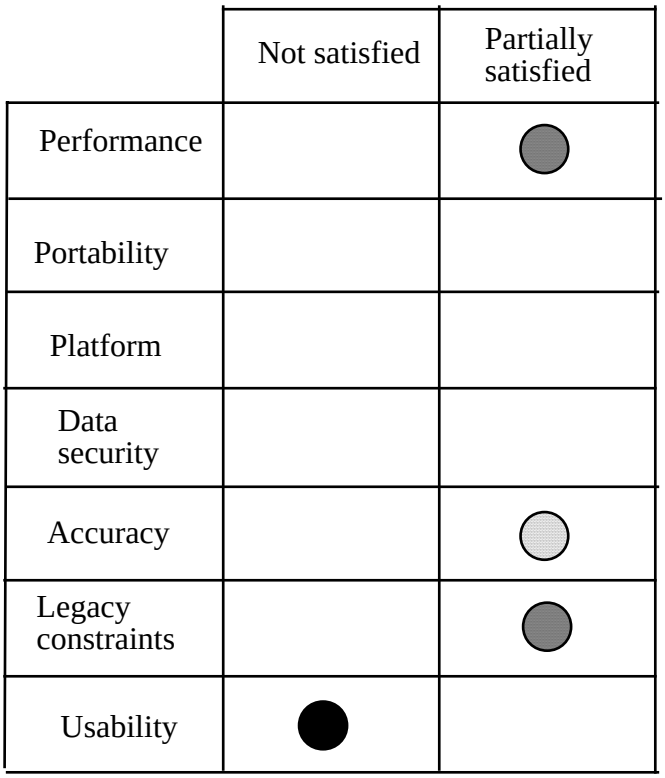

Figure 5 Implementation of functional requirements; the shading denotes the extent of the deficiency in the implementation.

The system shall retain the core functionality of the legacy Cobol system HIPPOCRATES and also cater for new requirements. Data formats shall be retained for database compatibility. The constraint of forward compatibility of data structures was not met and this caused considerable problem in transfer of history data.

- Usability

The interface should be usable by inexperienced users after two hours training. Most of the system functions should be usable without looking at a manual. The consistency of the interface across the application should meet the usability requirements stated above. This non-functional requirement was not satisfied, as demonstrated by the large number of usability errors.

4.4 Requirements implementation summary

In spite of the fact that many requirements were explicitly stated in the tender documents and specified in detail by the legacy system, there were several cases of omissions and unsatisfactory implementation.

Not Implemented. Only 2 information requirements fell into this category. The requirements were identified during the requirements stage of the project, but design solutions failed due to technical shortcomings. The information was mapped in the database schema, but could not be retrieved by the user interface. The problem lay in the interpretation of the requirements by the development team, who argued that these requirements were not clearly stated in the documentation that they had received.

Partial Implementation - Poor design. A number of the missing functionality errors encountered were not clearly specified in the initial requirements documentation. They were left out of the design because either the lack of formal documents during meetings (minutes), to record decisions were taken or poor user involvement in the RE process so that details were not correctly specified. In addition the 11 usability problems fall into this category. Lack of early validation of the user interface lies at the heart of the usability problem. User requirements that were not explicitly requested by the HIG analysts were revealed through semi-structured interviews, as follows:

- The requirement for a drill-down search facility by clinic. This was dropped during negotiation by the analysts of the Ministry as it was felt that its implementation was technically complex. This requirement was also reported as an observed problem (Error 11). It should be noted that due to the lack of proper negotiation with users, the latter were not aware of the fact that the requirement had been excluded from the specification documents.

- The requirement to provide information on expenses/revenue per clinic. This was also dropped during negotiation by the analysts as it would necessitate organisational changes, since hospitals do not keep such details.

4.5 Newly evolving requirements

Some new requirements were identified during user testing and from interviews. Given the short time period since implementation more requirements may be expected to evolve in the future.

- The requirement to provide information on waiting times for operations. This was a requirement that emerged later in the process. To meet budget and time restrictions the development team decided that new/evolving requirements would only be dealt with as extensions of the new system.

- Equipment and supplier data. This requirement emerged during development but was not implemented because of pressure on delivery deadlines

The designers also included some features because they felt that these are the state of the art of new technology, without having been asked to do so. Error 3 is an example where a 'drag' facility was implemented for the columns in configurable reports although this had not been asked for by the HIG analysts. An automatic layout should have been designed in light of experience with the previous version of HIPPOCRATES.

5 Process history and Problems encountered

This section describes the activities that took place in the RE process. The analysis adopts the high-level process model for RE proposed by Sutcliffe [9] as a framework for investigation. The system development followed the traditional life-cycle proposed by structured systems development methods, although it did not follow any one specific method. The impact of the IBM method was minimal as the project management guidelines were not followed in this project.

Feasibility

Problems contained in the legacy system and the availability of new technology were the reasons underlying the need for change. System evolution towards an improved version of HIPPOCRATES was identified as a solution to the user problems; however, organisational issues concerning the new roles of users and developers were discussed.

Scoping

The scope of the new system was set largely by the legacy system. This activity was performed by the analysts of the HIG in collaboration with the users. Scoping the system also included finding the fit between the new technology and the existing work practices. The terms of reference were to provide a high quality, Health Executive Information System to support the Ministry in health policy-making and evaluation.

Fact gathering

In order to elicit requirements, facts were gathered from the following sources:

Primary users (Ministry's Health Planners), Secondary users (Departmental managers and Ministry officials), Existing HIPPOCRATES software system, Literature about health care and executive information systems.

Facts were gathered from internal users using interview and observation techniques. The existing software system was also used as a starting point and much of the factual and development knowledge was reused. External (non Ministry) users were not formally interviewed. The analysts of the HIG used their experience concerning typical external user requests to describe their requirements.

Analysis

Analysis was based on the old system and the problems that had already been discovered after prolonged experience. The core functionality was carried forward to the new system. The shortcomings identified during the ten year use of the system were discussed. By analysing and reviewing current operations closely with users a more complete set of requirements was identified that included:

Refined summary and drill-down search facilities, data display using charts and graphical representation tools, more user-friendly interface based on GUIs and direct manipulation, integration of a data entry component.

Modelling

The system was modelled using entity relationship techniques which described all the major entity types and functional relationships. The process view of the system was not modelled.

The analysts did not perceive the analysis and modelling activities as being very critical to the development. Requirements were seen as modifications rather than implying the need for a completely new requirements specification. User involvement was limited. The idea for the new system came from the Informatics department and not from the user population. Therefore, users were not really involved in the process nor committed to the system. The staff of the HIG knew the domain very well and felt that they had a deep understanding of the system.

Validation

Validation was carried out by limited inspection, without formal walkthroughs, in meetings where the relevant documents were shown to internal Ministry users. For practical reasons no external users could participate in the sessions.

Having completed the analysis and specification of the requirements, the analysts proceeded with issuing a Request for Tender for which DBSoft made a successful bid. The HIG analysts provided the contractor with the specifications along with the legacy system database schema, the full set of manuals of the old Cobol system and its output, the annual book of health which contained samples of reports, and a demonstration of the old system. When the contractor had been determined the design and implementation phases of the project began in May 1995.

Design and implementation

After market research the development team tried to analyse the requirements from the tender documents and supporting materials. A second phase of requirements analysis was conducted between the HIG manager and members of the development team. The software contractor was required to prepare the specification documents for HIPPOCRATES. The project was given an optimistic deadline of six months. The DBSoft development team selected Powerbuilder as the software platform and the toolset they would use.

The developers built their own database schema and a data entry system in Powerbuilder so that they could access the database. After having completed this first phase of development, they started designing the user interface. At the end of October 1995, the development team sent a report for approval to the Ministry specifying exactly what the new system would do. Some minor comments on user interface features were made on that report and the implementation started, taking the comments into account. A first delivery for testing took place in the beginning of January 1996, two months after the initial deadline. The Ministry's analysts assessed the system and reported the following problems in detail.

- Poor system performance.
- Poor interface design.

- Missing functionality. Several Information Retrieval functions were missing, in particular:

  1.Operational data was not broken down by hospital sector and ward.

  2.Personnel data was not broken down by type of medical service, specialty and staff rank.

  3.Financial data was not broken down by revenue/expenses or general ledger categories.

  The development team treated most of the missing functionality as requirements discovered late in the process, which are not normally accommodated by fixed price contracts. The HIG analysts held a different view and the requirements were re-negotiated. It was decided that the development team would implement all the reported problems. The developers had to make changes to both the interface and the database schema, which is needed more time so a new deadline was set for June 1996.

  The development team performed informal evaluations of the user interface with developers not on the project who performed a few tasks and commented on the usability of interface features. Another prototype version of the system was delivered to the Ministry for testing in June 1996. The final operational system was delivered in the middle of July 1996, eight months late. At the current time of writing the system has not been finally accepted by the client. The company dealt with minor only problems leaving the major ones untackled and is requesting an extension of the contract to proceed. To make things worse the manager of the HIG who was committed to the project has been given new responsibilities. The prospects are that an agreement will be reached for completion of the project, however both sides have suffered from the potentially avoidable problems in the RE process. DB Soft have not been paid for their work and the Ministry continues to operate the legacy system.

  Process Meeting

  Meetings took place throughout the RE and development process. Two broad categories of meetings could be distinguished: first, meetings between users and HIG analysts to gather facts from users about their work, to capture new requirements, to brief users on the system specification and validate requirements, and training the users once the new system had been installed. Secondly, meetings between the HIG analysts and the development team to clarify ambiguous requirements and domain information; to negotiate and resolve conflicts over missing requirements and design options, and to review progress and test prototypes. The latter took place in DBSoft offices as there was no prototype system installation in the Ministry. Meetings occurred less frequently in the beginning of the process (monthly meetings) and more frequently towards the end of the process. During the last seven months weekly management reports were circulated between the members of the development team and the account manager of DBSoft in an attempt to prevent further deviations from the Ministry's specifications. These reports were also communicated to the Ministry. Some documentation was maintained throughout the project, although it was not detailed. However, the software

contractor did document intermediate versions of the system and established links between problem reports, produced in January 1996 and system iterations.

5.1 Problems encountered in the process

In this section the problems encountered in the process of developing the system are reported, with emphasis placed on RE problems.

5.1.1 Communication problems

Interpretation of requirements. Requirements were not correctly interpreted by the development team of DBSoft. Indicative of this was the fact that in January 1996 DBSoft delivered to the Ministry what they thought was a complete application for testing, which had a substantial number of shortcomings and several missing requirements. This problem was caused by poor consultation with the users and lack of domain experts in the development team.

Interaction of DBSoft with the Ministry and the users. Interaction was poor in the beginning of the process, when the requirements where being analysed and interpreted by DBSoft. The Ministry analysts had suggested weekly meetings, but meetings actually took place once a month. If there had been more frequent communication misunderstandings would have been discovered and responded to more quickly. Moreover, the development team had no interaction with the real users which might have helped in understanding requirements.

5.1.2 Social and organisational problems

Stability of the development team. Another factor that contributed to the implementation delay was turnover in DBSoft staff assigned to the project. New individuals were assigned responsibilities, consequently slowing the process down and making acquisition of domain knowledge even more difficult.

User participation. One of the most significant problems that both the HIG and DBSoft faced was the limited user participation throughout the process. Users were presented with a completed system which had not been approved by them. There was a history of lack of co-operation between the users and the HIG. Traditionally, all systems were built and operated by the HIG. Middle-aged user managers and administrators showed a lack of interest in Information Technology and most of them are still computer naive. The health planners who were interested in the project and eager to participate seemed too busy with their own work and it was very difficult to get their time and commitment.

Lack of mutual understanding. While the Ministry attributed many of the problems to the lack of understanding and the poor domain knowledge of the developers, the developers attributed the delay of the project to the slow response by the Ministry. The DBSoft project manager stated that 'a full documentation of the system under development was prepared and sent to the Ministry after the analysys had been completed'. However, the Ministry people did not highlight the severe lack of functionality until three months later.

Lack of trained personnel. The Ministry people also complained that the developers were not highly skilled and lacked the requisite experience in both Powerbuilder and Health Executive Information systems.

Different views on the system. Another problem was that the two parties had a different view of the system and placed emphasis on different system components. The Ministry analysts, being the representatives of the real users whom they had been supporting for years were mostly interested in the UI. DBSoft underestimated the importance of the interface and were interested in the underlying data structure and coding practices.

5.1.3 Politics

Politics were also apparent throughout the process. The development team agreed to implement the requirements that they thought were not included in the contract, in order to maintain a running professional relationship with the Ministry. Also, the development team stated that they had compromised their standard project management techniques, mostly because of the informal nature of the co-operation. This is strange considering the request by the HIG analysts for a frequent schedule of meetings.

5.1.4 Technical problems

The development team faced various problems during the data transfer, where the data from previous years could not be automatically transferred to the new system because of incompatibilities. Some data had to be re-transferred and part of the code had to be rewritten. The development team could not take full advantage of the features of the RDBMS system because the Ministry analysts disagreed with changing the underlying data structure and the database design. An example of this was the financial sub-system. DBSoft's view was that by sticking to the old system they were not making full use of the new technology. The Ministry's view was that the changes proposed would affect the logic of the program and attributed these suggestions to lack of understanding of their requirements.

Performance problems were discovered when the real data was loaded in the system during a prototype test. The problem was finally solved by fine tuning the system by creating indexes on database elements and creating an update facility for intermediary summary tables.

Overall the process appeared to be composed of two teams who communicated infrequently and who held different priorities in implementing the requirements. In spite of the development teams lack of consultation they did attempt to deal with the user's requests. The documentation from meetings was almost non existence so tracability of decision making during the process was not possible. Although the system requirements documentation, in the form of the old legacy system and new output requirements was reasonably complete.

6. Improvements to the Process

One of contributing factors to the failure of HIPPOCRATES is the lack of a systematic RE method. The developers did follow a standard systems analysis and design method (Solution 2000), however, methods of this gendre do not specifically address requirements analysis. Even if the developers had attempted to use a commercially available RE method, e.g. Dynamic System Development Method (DSDM 1995), they would have still been poorly served, as currently available methods do not address requirements in a legacy system context . This section proposes a RE method that attempts to deal with the legacy system problems and some of the difficulties encountered in the case study.

The basis for the method is the framework proposed in our previous work [9]. This pointed out an origin for RE in problems emanating from an existing system and proposed an outline process for requirements analysis and validation but did specifically deal with legacy system constraints in detail. The method we specify in this paper adopts the process model outlined in the conceptual framework. The main implication of legacy systems is the constraints they place on new requirements and the need to integrate changes resulting from new requirements without introducing errors into acceptable parts of the existing system. An overview of the method is given in figure 6. The stages are as follows:

Scoping

Establish the extent of change to the legacy system can be helped by scoping where new requirements will impact on a high level model of the existing system architecture. The architecture describe major system components, however, since legacy system change may not always arise from functional upgrading it is useful to consider where change may impact on a layered architecture, e.g.

user interface components, look and feel as well as task support functionality;

system functionality, as described in sub systems;

database and data structures;

host platform, network connectivity and distributed architecture.

Scoping also needs to be driven from a problem oriented approach, as the need for legacy system change is often motivated by perceived problems rather than completely new functionality. In HIPPOCRATES the scope of change impacted primarily on the user interface and the database, as well as upgrading the host platform. Early scoping requirements in this manner could have focused the development team on the key user requirement of usability.

Fact gathering

Legacy systems and their documentation provide a rich source of information for fact gathering. Only too often documentation is missing or inadequate so reverse engineering techniques may be necessary. Description of these techniques is beyond the scope of this paper. Information should also be captured from users and if possible the

original designers. Interviews and observation of the legacy system in operational may uncover many facts which were never documented. Use of the existing system is a rich source of facts and new requirements as users can volunteer facts in the context of using the system and point to problematic components on the user interface or demonstrate inadequate functionality.

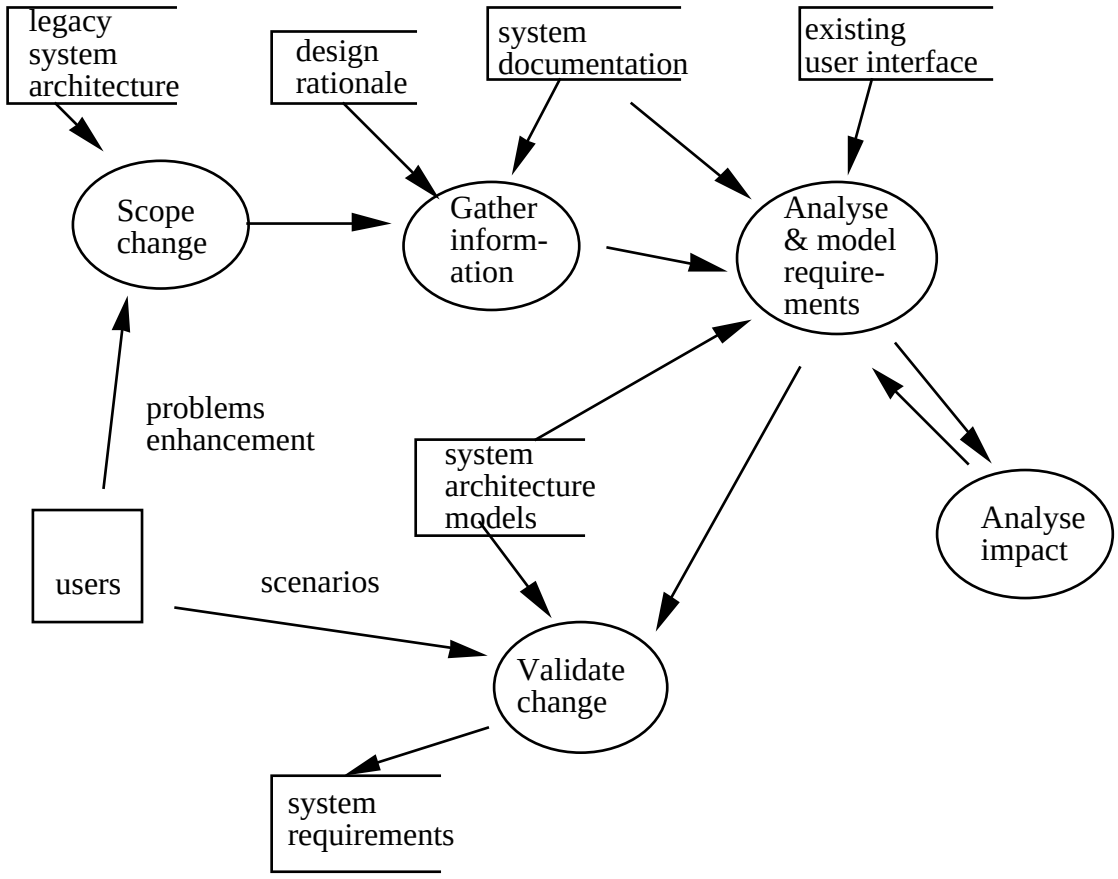

Figure 6 Method expressed in the DFD Format

Analysis

The system requirements specification may be developed from previous documentation, if it exists, otherwise system behaviour may be analysed by testing the interaction with typical scenarios acquired from users. Problems should be analysed to establish which components need to be changed or whether new system components will have to be constructed. The high level system architecture model constructed in the scoping phase should be refined so requirements for change can be located in specific parts of the system. Analysis may be driven from two viewpoints:

Stakeholders problems: in this case requirements have to be understood in terms of what part of the process is inadequate and why. Problems may be non-functional in

nature, e.g. performance or quality issues. These have to be traced to the functional/architectural components involved.

Environmentally driven change: requirements in this viewpoint are imposed on the system by the need to conform to other external systems, for instance interfaces to networked systems, new user interface standards or operating systems.

User enhancements: requirements originating from the users' wishes to improve the current system either because the social system it serves has changed or new technology offers new opportunities for supporting users in innovative ways.

Requirements are analysed to partition the necessary changes and ascribe their origins to particular stakeholders. In HIPPOCRATES most changes were motivated by environmentally driven desires to have a more modern user interface, although there were problems in information retrieval and presentation that needed to be fixed.

Modelling

Modelled is invariably interleaved with analysis. The format of models adopted may be determined by the legacy system documentation, for instance if entity relationship techniques were used to specify the database then it makes sense to use ER diagrams for new data structure requirements. In addition a system architecture model helps to locate change in specific components. Architectural models may be provided by the legacy system documentation or created afresh. An important aspect of analysis and modelling is establishing constraints and dependencies. Constraints are discovered by enquiring how new requirements or changes will be implemented. This often involves tracing dependencies between system components and design assumptions. Simple questions can be used with the system architecture model to assess the impact of new requirements. The key options to establish are:

Does the requirements entail development of a new module/component ? If so can it be integrated smoothly with the old system ? The module's inputs have to be traced to other system component or the user interface. Similarly output are analysed.

If the requirement entails modification of an existing module, how extensive is the change and what is the quality of the original design and software code. More extensive change and worse module quality indicate a complete re-design should be favoured. As before module connectivity has to be traced and interfaces designed.

Does the requirements entail changes to system interfaces ? These requirements originate from changes to platforms, operating systems, and other system software. Module interfaces have to be re-designed or inter-module bridges constructed.

In the case the major impact of the requirements was on system interfaces to upgrade the GUI, database and system software. However, there were changes to system functionality for data retrieval and failure to isolate these changes led to an inadequate implementation.

Validation

Validation is a key activity as changing existing system can have undesirable side effects. Validation in the sense of ensuring that the requirements meet with the stakeholders intentions can be addressed by developing concept demonstrators of the new system so users can test it see, alongside the existing system. Scenarios of use can be employed as test cases and run against either prototypes, concept demonstrators or story board mock-ups of the enhanced system to ascertain if the requirements meet with the users' approval.

Little validation was carried out until a fully functional prototype was shown to the users, when several modifications were suggested. These could have been anticipated by use of storyboards or simulation prototypes.

Verification and Impact assessment.

Verification usually implies formal modelling and prove of a requirements specification, however, this is frequently not possible because the legacy system software and documentation are not sufficiently detailed or rigorous enough to allow formal modelling. Nevertheless checking the impact of requirements on a legacy system is a vital activity that can be approaches with informal techniques. Each component in the system architecture can be checked to assess whether new inputs and outputs require:

(a) changes to algorithms, calculations or component functionality

(b) change to data structures

(c) altering the component's interface to other modules in the system architecture

Locating and assessing the impact of change leads to estimating the costs of implementing requirements and prioritisation of the changes. Some changes may have localised effects and hence can be implemented without too much risk. The effect of change depends on the design quality of the legacy software. Poorly design cod with high coupling and low cohesion will make anticipating the knock on effects of change difficult, let alone identifying the appropriate locus for change in software code. Impact assess, therefore has to take the feasibility of change into account. In many legacy system poor design and lack of modularity may make change impossible. Sometimes the client may want to retain and inadequate design for other reasons. In HIPPOCRATES this occurred when the users wished to keep the original database schema that they were familiar with even though it hinder implementation of new data retrieval requirements.

7. Discussion

The history of HIPPOCRATES can be viewed in two ways. On one hand only a few requirements were missed during analysis and most were implemented, so the system might been seen as a success. On the other hand, after a short trial period, was decided to give the system back to the developers to fix the usability problems. An agreement was reached with them to do the job with no extra cost, so the developers paid the penalty for poor requirements analysis. However, to date, they have not yet complied

with all the client's requests for modifications, so the project still runs the risk of being abandoned. The prime motivation to develop the new system was the need for enhanced usability, since the old system met most of the functional requirements in a satisfactory manner. The failure of the developers to appreciate that a non functional requirement, usability, was the client's main concern caused significant problems. Our investigation has uncovered many problems and dissatisfaction among the users that emanate from poor usability. The glaring communication failure between developers and users meant that usability was not seen as a priority. Consequently the system, although functionally reasonably correct, had many irritating operational problems that annoyed the users.

In addition to the non-functional requirements there were functional requirements that were not implemented, in spite of these defects being notified to the developers. This demonstrates the failing in communication and the penalty of not adopting iterative development with early requirements validation. Functional and information requirements were correctly captured by the Ministry analysts but were subsequently missed or misinterpreted by the software developers. User-interface requirements, which were not documented in requirements specifications, were missed because of the lack of awareness of the importance of usability and absence of user involvement in the design process, a common failing that can lead to ineffective and inefficient systems.

Several requirements were not asked for or dropped during negotiation because they could not be accommodated within the fixed price contract. Evolving requirements is a major issue and this study demonstrates that procurement contracts and poor communication present considerable barriers to their resolution. Methods that manage change, such as Dynamic System Development Method [12] may be one answer. The effect of different contract types in the RE process is another implication that has not been adequately addressed by research, although the process model for procurement of Ncube et al. [13] does suggest how flexibility may be increased to respond to requirements evolution.

The majority of problems faced in the RE process were non-technical ones, namely communicational, social and organisational problems and politics. This finding is concordant with the results of earlier studies of RE practices [3], [6], [7]. However, few reports have noted legacy system constraints on RE which caused significant problems, i.e. DBMS updating.

The lack of domain expertise in the software development team that contributed to misinterpretation of requirements was a major problem. This supports the findings of Curtis et al. [2] and the LAS inquiry [1], also agreeing with the findings of Lubars et al. [3], who noted that "we were surprised not to hear of any major problems caused by misunderstandings about the customers requirements or domain specific assumptions. The absence of domain related problems indicates that the projects possessed a high level of domain experts." Another problematic area identified in this study is that of the instability of the development team and the lack of trained and capable personnel. Given that previous studies of RE practices have found that personnel capability and knowledge impact on the success of the RE process [2], [3], this has implications for staffing and training practices.

Poor interaction of the software developers with the customer's analysts and users was also a problem, especially in the early phases of development. This is also noted by Waltz et al. [4] and Lubars et al. [3]. In particular, lack of negotiation and validation with users is a significant problem since user-involvement in the RE process has been supported by many authors. The adoption of methods that encourage user participation are obvious cures for this problem, e.g. ETHICS [14], RAD/JAD workshops [15], and Co-operative Requirements Capture [16].

HIPPOCRATES started with a legacy system, so it represents a particular type of requirements engineering exercise. Sutcliffe [9] points out that the starting point for RE and the application type can have important implications for the process. This case study was requirements by example for a bespoke system developed by contractors. This gave the requirements engineers a head start because the legacy system represented a near complete requirements specification. They also had access to copious system documentation. Furthermore the change from the legacy system in terms of its basic functionality and information content was not radical. Given these advantages we might expect most of the functionality requirements to be captured and successfully delivered. What this study has shown is that for systems of this type, satisfying non-functional requirements is the main problem, in particular usability. What is also interesting is that the ambition for HIPPOCRATES was modest; the users, for instance, did not develop requirements from benchmarks of leading health EIS's in other countries. Although the developers tried to suggest limited technical innovations these were not well received.

A final reflection is that this study has shown that systems can be developed in spite of severe deficiencies in the RE process. We discovered most of the classic problems in poor communication, lack of validation, user participation, etc. Nevertheless the development at least partially succeeded, in contrast to the failure reports of large scale systems [1]. Some reasons may be that RE is less demanding in information systems with limited functionality, modest size and a legacy system to demonstrate baseline requirements by example. However, we note that the system could have been far better and our study points our process improvements that could have been made; furthermore, it demonstrates that failure to attend to a single (in this case non functional) requirement can have serious consequences for system acceptance. Investigating current practices of RE and identifying problems encountered in a forensic manner can yield significant conclusions, and supplement the general observations of El Emam and Madhavji [6], who do not give insights into the pressing question of how and why requirements actually get missed or mistaken. This study went further to diagnose the causality of requirements engineering failure, but it reports only one system. Many more empirical studies of this nature are required so we can better understand the causes for failure in the RE process that lead to problems in different types of applications.

Acknowledgements

The authors would like to thank the staff of the Health Information Group and users in the Greek Ministry of Health, and personnel from DBSoft, Athens for their help in this study. This research has been partly funded by the European Commission ESPRIT

21903 'CREWS' (Co-operative Requirements Engineering With Scenarios) long-term research project.

References

1.HMSO. Report of the inquiry into the London Ambulance Service. HMSO, London 1993.

2.Curtis B, Krasner H, Iscoe, N. A field study of the software design process for large systems. Communications of the ACM, 31 (11). 1988. pp 1268-1287.

3.Lubars M, Potts C, Richter, C. A review of the state of the practice in requirements modelling. First IEEE International Symposium on Requirements Engineering, San Diego, California. IEEE Computer Society Press. 1993. pp 2-15.

4.Waltz D, Elam J, Curtis B. Inside a software design team: knowledge acquisition, sharing and integration. Communications ACM 36 (10). 1993. pp 62-77.

5.Timpka T, Johansson M. The need for requirements engineering in the development of clinical decision-support systems: a qualitative study. Methods of Information in Medicine, 33, 1994. pp 227-233.

6.El Emam K, Madhavji N. H. Measuring the success of requirements engineering processes. Second IEEE International Symposium on Requirements Engineering, York, 1995, IEEE Computer Society Press. pp 204-214.

7.Al-Rawas A, Easterbrook S. Communication problems in requirements engineering: a field study. Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London, 1-2 February 1996.

8.Fischer G. Beyond Human Computer Interaction: Designing useful and usable computational environments. In People and Computers VIII, Proceedings of HCI'93, Ed Alty J.L et al, pp 17-31, 1993.

9.Sutcliffe AG. A conceptual framework for requirements engineering. To appear in Requirements Engineering Journal, 1997, 96/12. Centre for HCI Design Report, School of Informatics, City University, London, 1996.

10.Monk A., Wright P, Haber J. and Davenport L. Improving Your Human Computer Interface. Prentice Hall, 1993

11.Sutcliffe, A.G., Ryan M., Springett M.V. & Doubleday, A. Model Mismatch Analysis: Towards a Deeper Explanation of Users' Usability Problems. Centre for HCI Design Technical Report, School of Informatics, City University, London, UK. 1997.

12.DSDM Consortium. Dynamic Systems Development Method. Tesseract Publishers, Farnham, Surrey 1995.

13.Ncube C and Maiden N.A.M., Procuring Software Systems: Current Problems and Solutions, to appear in Proceedings of REFSQ workshop, Barcelona, June 1997

14.Mumford E. Effective systems design and requirements analysis: the ETHICS approach. Macmillan. 1995.

15.Martin J. Rapid Application Development. Macmillan. 1991.

16.Macaulay LA. Requirements capture as a co-operative activity. First IEEE International Symposium on Requirements Engineering, San Diego, California, 1993. IEEE Computer Society Press. pp 174-181.

17.Markis P. HIPPOCRATES: an On-line Interactive Health Data Retrieval and Processing System. Medical Informatics, 9 (2), 1984, pp 79-91.

Appendix. Usability errors

Poor/absent feedback

E12.In the health institution selection screen the Select icon offers no feedback of whether the selection has been made. When the users click the icon a short message (’please wait’) appears, but no subsequent message confirms that the system has completed the action.

E13.The type of health institution that has been selected is not displayed in most subsequent screens. Therefore, the users often forget their selection, especially after an interruption.

Misleading cues

E14.The system allows for only nine data elements to be displayed; the users are not warned when this number is exceeded and no data appears on the screen. This is poor error prevention that increases user work.

E15.The meaning of the two icons in the bottom left-hand side of the health institution selection screen is ambiguous. Most users were expecting a single OK button to perform the selection. The existence of a Get Info and a Select icon is misleading and there were individual differences in interpretation.

E16.In the screen where health indicators are created the Enter icon is ambiguous. Once users had typed the indicator in the text box provided they did not know how to proceed. The existence of some clarifying text associated with the image would be a possible solution.

E17.The system does not prevent functions that are inappropriate. The Print icons are active even when printing is not possible. The icons could be ’greyed out’ during certain stages in the dialogue to show that they are not selectable.

E18.The Up and Down Arrow icons are redundant. They perform the same function as the scroll bars in the display windows. Moreover, the users found it difficult to use them because they had no immediate meaning.

E19.The icons used to start processing and to go forward one step are different in different screens. In some screens the Get Info icon is used and in other screens the Magnifying Glass icon is used. This lack of consistency increases the learning load for the users.

E20.Selectable areas or elements in the screen were sometimes shown by a ’pointing hand’ and at other times by an arrow. This caused inconsistency and as a result the users were misled into thinking that a certain area was not selectable although it was.

E21.The hourglass is not focused. The users were misled into thinking that an operation was complete and tried to proceed to the next action.

E22.Many users were confused by a Go Back and an Exit icon in the same screen. The inclusion of both icons in the same part of the screen was misleading.