INNOVATION OF A DESIGN METHOD (MoIST) THAT INCORPORATES NON-TRADITIONAL 'SOFT' SYSTEMS SCIENCE INTO TRADITIONAL 'HARD' INFORMATION SYSTEMS DESIGN

JUDITH A. NWAKO

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The University of Huddersfield

July 2007

Acknowledgments

To my beloved husband Sekao Nwako – You are an amazing gift, an exceptionally excellent man and the best husband.

To my wonderful parents Leon and Grace Hopkins and my siblings Ava, Sonya & Kevin and all my aunts, uncles and cousins – Thank you for always loving, encouraging and believing in me.

To my pastors Dr Ramson & Linda Mumba and Dr Jerome & Ruth Anekwe –
By a prophet he brought them out and by a prophet he preserved them.

To Dr Ian Pichford & Carol Doyle – Research Office and to Dr David Hall, Dr Steve Wade, Dr Steve Littlewood & Dr Stephen Catterall – School of Computing & Engineering. Thank you so very much.

This is the Lord's doing and it is marvellous in our eyes and to God be all the thanks and all the glory!!

Abstract

The research explores ways of making the information systems development process more effective. The thesis documents an original design method. This is the method of incorporating Systems thinking into Information Systems Design (MoIST). The thesis demonstrates that MoIST improves information systems design and adds to the effective arsenal of methods that already exist.

The Computer Science literature has identified some weaknesses in the software development methodologies. These weaknesses include premature design decisions taken before major requirements are known. Another is the dearth of options for applying Systems Science and information systems design techniques in a UML-based context. It was found that these weaknesses sometimes resulted in software failures. These findings have been confirmed in the empirical and the evaluation portion of the research.

The essence of the thesis is that appropriate software development strategies may be chosen at various points in a project. The choice of strategy is based upon the value of particular factors. These factors include confidence in requirements, development environment structuredness, user types and developer types.

In order to achieve the research aims, the MoIST is utilised to preserve the methodological strengths of the hard systems engineering paradigm. It simultaneously attempts to minimise its weaknesses by combining it with a systems science approach called Soft Systems Methodology (SSM). The research incorporates this non-traditional 'soft' Systems Thinking into traditional 'hard' Information Systems Design. The two main contributions of the thesis are the transformation of SSM conceptual models into UML use case diagrams and activity diagrams. Another is the creation of MoIST Project Option Selection Tool (MoPros).

This MoIST method has been tested empirically by utilising it in a complex, unstructured setting in a School of Computing and Engineering. Based on the theoretical and practical work conducted, it is concluded that the MoIST method is effective in several ways. It provides coherence and structure to complex software projects and can help to facilitate decisions about improvement strategies. It also successfully incorporates the results of SSM analysis into requirement specification

based on the UML. The MoIST method is offered as a viable option to add to the existing development alternatives for successful software development.

Table of Contents

LIST OF FIGURES AND TABLES

Chapter 1

1.1 Introduction

In many nations of the world today, news of instability and many disasters are shocking happenings being reported across the globe. Economic reverses, terrorism and fear seem to have become the order of the day. In stark contrast to such negativity, the astoundingly positive technological progression of our day would have boggled the minds of our forefathers. They would not have believed it then even if they were told (Newsweek, 27th January, 2005).

As a microcosm of global changes, some universities in the United Kingdom (UK) have been experiencing radical changes and "shake-ups". This has led to the rethinking of some of the teaching and learning strategies currently employed in these universities (Wend, 2004). In the area where I lived and conducted my research – the north of England – schools and departments within universities have been similarly affected. Having spent three (3) years in my particular School of Computing and Engineering has afforded me a unique perspective on the organization. This has led to a greater familiarity with its attendant virtues and unstructured complexities. This vantage point is not easily gained by most external consultants, developers or methodologists. Being an insider provided me with multiple advantages and benefits. These included a more comprehensive understanding of the real functioning of the organization and made the organization's cultural ethos more transparent. This made it possible to effect a more successful and relevant intervention in the organization as I was in the midst of the action research being carried out. One lesson I learnt from this research is the value of staying very close to the phenomenon one is studying, rather than doing scholarly work at arm's length (Sankaran, 2001). Some of the difficulties experienced by universities are the result of the changing times we live in. This is exacerbated by national governmental guidelines, international, educational, political and cultural dynamics.

In recent years the UK government has indicated that Widening Participation in higher education is among its highest priorities.

On the 22nd of January 2003 the Secretary of State for Education and skills, Charles Clarke, announced publication of the white paper "The future of Higher

Education" which sets out the government's plans for radical reform and investment in universities. Participation in Higher Education will equip people to operate productively within the global knowledge economy. It also offers social benefits, including better health, lower crime and a more tolerant and inclusive society. It aims to ensure that all those with the potential to benefit from Higher Education have the opportunity to do so, whatever their background and whenever they need it (http://www. hefce.ac.uk).

Ambitious targets have now been set for universities. This is to ensure that the advantages of a university education are available to as wide a constituency as possible. The government seeks to rapidly increase the number of students from traditionally underrepresented social and ethnic groups. The government stated its intention of aiming for a 50% participation rate in higher education (DFes, 2003). This has left universities debating whether to lower established standards to ensure high recruitment levels or whether to spare no cost in getting students up to the requisite academic level (May and Bousted, 2004). In the midst of these governmental strategies, retention of students is at an all-time low (Webb and Hill, 2003). Several educational reforms have been recommended that affect secondary, further and higher education. These include the 14-19 Tomlinson's Report done by Mike Tomlinson on the state of secondary education in England, the Dearing Report, the National Curriculum reform among others (DFes, 2003).

During my time of research at the University, the School of Computing and Engineering experienced some major changes. Several years ago, the popularity of Computing as a seemingly lucrative career choice attracted many new students. Subsequently there was a great rise in the student numbers. This resulted in new staff members being employed to meet the demand. Within a matter of years, retention of students dropped to approximately fifty percent (50%). This meant that there was a dip in the financial viability of the school. Consequently, approximately fifty percent (50%) of staff were made redundant. This was not an isolated incident which was peculiar to one university, it is a reflection of happenings in the wider society and is replicated in universities all over the UK (Education Guardian, 19/10/2004).

Universities have been left by governments to be largely self funded. Their financial viability and existence depends largely on their valued clients – the students. Recruitment and Admissions are areas that universities are attempting

to streamline in order to become more efficient. As Professor Stephanie Haywood, the Head of Engineering at the University of Hull, UK said,

'we all feel the squeeze these days. Our student numbers are good but our per capita grant is decreasing year on year in real terms, so if we want to maintain the size of the department, the pressure is on to bring in new income'

(Education Guardian, October 19, 2004. page 20)

It is a common practice today for policy makers in organizations to attempt to regulate this kind of problematic situation by economic means and by mechanisation using Information Technology. This regulation is exercised at the expense of other types of regulations. These are namely social, emotional, political and cultural aspects of organizations. These are as important to organizations as economic factors; yet these are largely ignored in most modern organizations. Organizations generally and in this specific case, universities, are expected to function as businesses (Mirijamdotter, 1998). This is usually the prevailing modus operandi of technocrats in organizations. As problems occur, the perception is usually that there is not enough time to 'waste' thinking about the problem and structuring or formulating a solution. Instead, an immediate technological solution is sometimes proposed. This is the bias towards hard systems engineering solutions. This is not inherently incorrect or illogical. The problem though is that these hard systems solutions have limitations and therefore do not always address the root problem (Ulrich, 2003). This mechanistic view has generated criticism from some quarters of mainstream Computer Science. It has subsequently led to the development of new fields such as Systems Thinking, SSM, Business Process Modelling and Critical Systems Thinking among others (Checkland, 2000, 1998, 1981, Ulrich, 2003).

The School of Computing and Engineering and its areas with potential for streamlining has been the focussed area of this research. In order to make some sections of the school more efficient, 'hard' software systems such as Blackboard v 6 - a Virtual Learning Environment and Attendance Monitoring Software were introduced to help bolster the retention levels. To date, these hard solutions, whilst effective in their own right; seem not to have prevented the continual drop

in student rates. Logically, this suggests then, the possible existence of other solutions or more accurately phrased – additional solutions to hard systems.

Soft and hard methodologies cover different parts of the life cycle, particularly when there is uncertainty about the goals or strategy of the organization as a whole. A hard approach will be more appropriate once any initial uncertainties and ambiguities have been resolved (insofar as this is possible), since the emphasis then shifts to a specific project with relatively clear goals and boundaries… in certain situations, hard and soft methodologies can complement each other, and can be used together to help overcome some of the perennial difficulties in systems development (Bennett et al, 2002 p 568)

This research shows that a valid solution lies in the utilisation of Soft Systems Methodology (SSM). The novel Method of incorporating Systems Thinking into Information Systems design (MoIST), developed in this research has been applied in the School of Computing and Engineering. This application of MoIST has been made to two (2) major functional areas. These are the Academic Skills Support Process and the Postgraduate Project Process. It is within this context that this thesis has been evaluated empirically.

1.2 Background to thesis

In the earlier years of software development, it was usually understood by IT practitioners that the phases of the Software Development Life Cycle generally led from analysis of user need, to requirements capture, to systems design, to implementation. Times however have changed and the software industry and its methods and practices have changed right along with it in sometimes radical and unpredictable ways (Henderson-Sellers and Unhelkar, 2000). As the software industry hurtles forward into emerging technologies in an increasingly complex market place, there is an urgent need. This need is for a flexible method that can function within a 'process environment' and can be tailored. This would be a customized method that would enable the software development process to be tailored to precisely fit the individual organization's development environment (Henderson-Sellers and Unhelkar, 2000).

The SDLC represents a sequence of stages in which the output of each stage becomes the input for the next. This cycle has been the bedrock of the software development process over the years since its inception (Centre for Technology in Government, 2005).

The SDLC's workings involve cycling through the phases of analysis, design, implementation, testing, deployment and maintenance. Over the years many amendments have been made to this sequence and the format; but the re-worked methods usually adhere basically to the tried and tested traditional software development life cycle. These include the reality that real projects rarely follow the sequential flow of analysis through to design and eventual maintenance. At the beginning of most projects, there is usually often a great deal of uncertainty about requirements and goals. It is therefore difficult for customers to identify these criteria on a detailed level. An effective software development method must deal with this natural uncertainty in an efficient manner. A software development project generally goes through the software development process more than once. The architecture is not always excellent and easy to use. The implementation design is not always sound. The realisation is not always fixable as testing proceeds. Mistakes might not be all in the realisation. Consequently their repair is not always smoothly interspersed with component and system testing. Generally one does not build a whole system all at once. Additionally, developing a system can be a long painstaking process that does not yield a working version of the system until late in the process (Centre for Technology in Government, 2005).

Times have changed. Software projects have gotten progressively larger as clients have become more technically savvy and knowledgeable (Online Software Development Magazine, http://www.sdmagazine.com, 2005). The dawning of the information age has empowered the typical computer science lay person to understand concepts that would have daunted others only a few years ago. Client expectations have dramatically increased over the years. This means that there is almost an abnormal demand for software product delivery in less time than formerly requested. This demand has spawned a proliferation of other noted methods (Henderson-Sellers et al, 2000).

The software industry still tries to meet client demands. Consequently year by year there have been numerous instances and reports of software failures. These failures have now become a badge of dishonour for software development as an industry

(Neumann, 1995; Ewusi-Mensah, 1997;Ewusi-Mensah, 2003). Many attempts have been made to ensure that the discipline of developing software is made more precise. The label of 'software engineering' was even coined to somehow convey the idea of the traditional engineering discipline with all its precision and super accuracy (Wilson, B, 1990). This change did not have the intended effect as it has been seen that software development is different from traditional engineering. Traditional engineering assumes that the problem is known while with software engineering, the problem to be solved has to first be discovered. In traditional engineering, the requirements are already clear and well known before the project starts. The client knows exactly what is wrong and as the requirements are well known, the engineering principles are then applied usually to a successful conclusion. In contrast, the requirements for software engineering are usually not known and must be discovered before any engineering principles can be successfully applied (Neumann, 1995).

Studies have revealed that the scope, complexity, and pervasiveness of computer-based and controlled systems continue to increase dramatically (Pullum, 2001). The consequences of these systems failing can range from the mildly annoying to catastrophic, with serious injury occurring or lives lost, human-made and natural systems destroyed, security breached, businesses failed, or opportunities lost. Software faults may be traced to incorrect requirements where the software matches the requirements, but the behaviour specified in the requirements is not appropriate (Pullum, 2001). This means therefore that the main focus for improvement must of necessity be focussed around those analysis and design phases of software development.

1.2.2 Thesis Aim

The thesis seeks to establish a stronger link between the requirements analysis and the design phases of the software development life cycle. This is achieved by extending and strengthening the requirements analysis phase using the Method of incorporating Systems Thinking into Information Systems design (MoIST, chapter 5). The MoIST method uses Soft Systems Methodology (SSM) to facilitate a comprehensive examination of the situation before it is modelled (Checkland, 1981, Checkland and Scholes, 1990).

MoIST joins the company of software development methods already being successfully used. These include ETHICS, BASE and BOOST (Mumford, 1993, 2000, 2003, Bustard, 2001, Dobbins, 1999). Each of these methods has its own way of meeting the needs of a variety of problem facets within the software development spectrum. MoIST's unique assertion is that if a particular software project matches the criteria stipulated in (see MoIST's Project Option Selector template, chapter 5) then, the MoIST could help to alleviate many of the eventual software difficulties encountered and increase the chances of a successful software product. This would utilise the generic principles of problem/stakeholder domain analysis and analyst/requirements engineers' domain analysis.

1.3 Deficiencies in existing requirements analysis methods

Effective software development should centre around establishing what the software system is intended to do. Solutions lie in the area of identifying ways of verifying what the system should do, what the client wants the system to do and what the system should do based on the developer's expertise (Si Alhir, 2003). When used alone in the requirements elicitation process, hard systems engineering models can encourage early design decisions before opportunities for improvement have been agreed (CCTA, 1993). Conversely, Soft Systems Methodology (SSM) used on its own in the requirements elicitation phase may lack some of the detailed information required by programmers. The literature clearly demonstrates that there are many advantages and some disadvantages in combining soft systems science with hard systems engineering. (for a more detailed review, see chapter 4). The research focuses mainly on the advantages of this amalgamation. It demonstrates ways of integrating techniques from SSM (Soft Systems Methodology into the requirements elicitation stage of a software system development method based on the Unified Modelling Language (UML) (Siau and Halpin, 2001; Booch et al, 1998; Fowler and Scott, 2000; Maciaszek, 2001, Si Alhir, 2003).

Use case analyses bring out very little of what goes on within a business, but they contain much more operational, logical detail than equivalent SSM models. This can help the analyst to better understand each activity. Through multiple perspectives, SSM promotes a truly thorough examination of why a business exists and hence helps to more fully identify critical logical decisions. By linking some of these softer approaches to the UML, it is then easier for the software practitioner to select a particular course of action in coming to terms with the problem. (Donaldson and Jenkins, 2001)

The deficiencies in the requirements analysis process has led to many documented software failures that will be discussed below (Abdel-Hamid and Madnick, 1990).

1.3.1 Resultant software failures

A project that fails to meet a customer's need, no matter how technically sophisticated or perfect it is, is destined to fail (Remenyi, D, 1999). One of the most consistent features of information systems development over the years has been its many failures (Ewusi-Mensah, 2003). These failures have ranged from complex, real time and life critical software systems to less sophisticated information systems. Examples include the London Stock Exchange Taurus System which cost approximately £480 million (Drummond, 1996; Willcocks and Graeser, 2001). This system was never completed or delivered. The verdict was an inadequate match between the needs of the users and the proposed systems solutions (Willcocks and Graesar 2001). Information Systems failures generally occur because goals and requirements are poorly defined.

These failures of conceptualisation involve either a misunderstanding of the clients' needs and requirements or a misunderstanding of technology. (Remenyi, D, 1999).

There is usually a failure to manage people and results in communication breakdown. Information systems projects in general continue to fail at an unacceptable rate (Abdel-Hamid and Madnick, 1990, Myers, 1994). Over the years much research has gone into finding more efficient means of building software systems and of ensuring that they are satisfactorily completed to the users' specifications. Methodologies for software have been developed and amalgamated over the years in order to aid in the production of successful computer systems. Despite this gargantuan effort, intensive research in the past has generated too little understanding of how to avoid failures in systems development initiatives (Mitev, 1996, Rosenwein, 1997). From the growing incidence of failed projects, (see table 1.1) it can be concluded that advances in technologies are not sufficient to save systems projects. Instead, they remain susceptible to failure until it is understood how technological, organizational, and institutional changes are interwoven in the systems and how systems developers should accordingly state and manage requirements for such systems Abdel-Hamid and Madnick, 1990; Myers,1994; Drummond 1996; Mitev 1996; Rosenwein,1997, Dittrich, Floyd and Klischewski,2002.

	Type of abandonment		
Published study	**Total (%)**	**Substantial (%)**	**Partial (%)**
Standish Group (1995, 1998)[*]	31 (28)[*]	52.7 (46)[*]	Not available
KPMG (Cole 1995)	10	28	24
OASIG (1996)	40	25	80
Ewusi-Mensah and Przasnyski (1994)	44	16	9

[*]Reported in Whiting 1998

Table 1.1: Summary data on software project abandonment, Ewusi-Mensah, K, 2003, Chapter 1

[1]Estimations on IT expenditure in the UK public sector for 2003/2004 alone have set it at greater than £12.4 billion. The projected sum for overall IT spending in the UK is deemed to be £22.6 billion. Despite these staggering figures, the majority of software development projects repeatedly fail to deliver crucial benefits in a timely fashion and fail to meet cost and specification targets (Report of the Royal Academy of Engineering and British Computer Society, 2003). Some of the notable ones are examined here.

Examples of Software failures

On Thursday, June 3, 2004, Thousands of airline passengers were left stranded when the UK's **National Air Traffic Control** computer system run by a thirty year old software crashed. Overnight testing of an upgrade of the flight data processing system precipitated the disaster. The software is not due to be replaced until 2010. The new National Air Traffic Operations Centre was slated to open in 1996. It opened in 2002, five years late at a cost of £623 million, twice its original cost. The plan is to spend £1 billion upgrading the system over the next eight years (BBC News, July, 8, 2004, Computer Weekly, June 8, 2004, The Independent, June 4, 2004).

[1] Report of the Royal Academy of Engineering and British Computer Society, April 2003

In 1999, **Highmark**, a US health insurance company contracted KPMG Consulting to develop an electronic billing and accounts receivable system (Ewusi-Mensah, 2003). The completion date was slated to be 2002. Two years later with the project only 20 percent completed, KPMG requested $8 million in addition to $12 million already paid to them. Highmark refused and sued KPMG.

The **Libra IT** system for magistrates' courts was commissioned in 1998. £184 million was paid out initially. Things came to a halt in 2002. The deal was revived in 2002 at a cost of more than £318 million. The total system development time jumped to 8.5 years. In 2003, Libra was labelled as 'one of the worst IT projects ever seen' by the chairman of the Public Accounts Committee (Report of the Royal Academy of Engineering, 2003).

Hershey Foods hired four (4) different consulting firms to work with its IS department in 1996. This was to work on the Enterprise Resource Planning Systems. In 1999, the implementation was three (3) months behind schedule. The decision was taken to condense the implementation time from the estimated four (4) years to thirty (30) months. This caused chaos. Hershey Foods took a year to fix all errors and return to some semblance of operational normalcy. If it were a smaller company with less resources, it might have never recovered, but might have been bankrupt (Ewusi-Mensah, 2003).

The **National Health Service** (NHS) Direct[2] is among the strategic information systems exemplars and beckons as a beacon of hope (see table 1.2). NHS Direct's mandate is to provide up to date health information. Another part of its mandate is to dispense advice to the populace of England and Wales via a telephone line and online facilities. Development began in 1997. The implementation timeframe for the telephone helpline was 2000. This was achieved in 2000. The target time for the online facilities to go live was set for 1999. This was successfully done in 1999. Success was attributed to effective use of piloting and wide ranging consultation of key stakeholders within the constraints of the tight schedule.

Ariane 5 had her maiden flight on June 4, 1996. This flight ended with the launcher exploding owing to a series of software failures.

[2] Report of Royal Academy of Engineering , 2003

The **London Ambulance** Systems failed twice in 1992. This was attributed to several software engineering failures. Specifically project management defects were cited as contributory factors. The monetary cost of the failure was relatively small at £9 million. The human cost however was much higher. This was because it was thought that many persons perished who would not have if ambulances had been on the scene in a more timely manner.

For **Therac 25**, between 1985 and 1987 persons suffered serious radiation overdoses. This was caused by software-related malfunctions of the Therac-25 radiation therapy machine. An important core cause was a lack of quality assurance. This led to an overcomplex, inadequately tested, under documented system being developed. Additionally there was failure to take corrective action.

In the **Denver Baggage handling** system fiasco, the system overran the planned schedule significantly. This prevented the airport from opening on time. The system had major software viruses and cost almost an additional $200 million more to ensure that it worked.

Taurus was a projected automated transaction settlement system for the London Stock Exchange. The project was cancelled in 1993 after being worked on for more than five (5) years. The project cost was approximately £75 million. The eventual cost to customers was estimated at £450 million. The credibility of the London Stock Exchange suffered immensely as a result.

The software failures point to the need to rethink the way software development is currently done. There are many reasons given for all of the above software failures. The root cause of all the failures however were linked to a poor understanding of the problem domain in each situation. There are many other factors that impact the software development process that need to be examined before the design and implementation occur. The view of software as a purely objective product that is created devoid of context is somewhat misleading. Context be it organizational, cultural, managerial or technological shapes the software development process and the final product created (Ewusi-Mensah, 2003).

1.4 The nature of the thesis solution

The search for the perfect, most efficient way of growing software and of software development has been refined and experimented with. Yet in spite of this, huge software developments continue to fail often and at great cost to taxpayers and corporations. (see table 1.1) A proliferation of methods and methodologies abound which purport to offer the cure for all software development ills (Henderson-Sellers et al, 2000). There has been enough evidence however to prove that there is no one set way of approaching software development. Many factors affect the appropriateness of a methodology, including the type of project (large, small, routine or mission critical), application domain (real-time, safety critical, user centred, highly interactive, distributed or batch mode) and nature of information systems development organization (Bennett et al, 2002, p 567). Each development project has its own unique set of heuristics and problems and its own set of unpredictable team members. What the software development industry needs then is not so much **a** perfect method for developing software; but an increase in the number of proven methodologies that work. (Report of the Royal Academy of Engineering and British Computer Society, 2003). These would add to the toolkit of developers and enable them to more closely match methods more closely to the characteristics of their projects. That there is still a demand for other software methods and tools is confirmed by a recent study.

> Further developments in methods and tools to support the design and delivery of IT projects could help to raise success rates.
>
> (Royal Academy of Engineering and British Computer Society, RAE & BCS 2003)

This research demonstrates how Soft Systems Methodology (SSM), a problem structuring methodology is integrated with a method based on the Unified Modelling Language (UML), an object oriented notation and graphical language. This results in an amalgamation of hard systems thinking and soft systems thinking. This fusion could possibly help to minimise many of the documented software system disasters and could utilise time, money and machine resources more efficiently. The Unified Modelling Language (UML) is relatively new and is filled with multiple possibilities for research. It is gaining much support as it is backed by the Object Management Group (OMG). The UML has become a leader worldwide and forms the basis of most modern software development Computer Aided Software Engineering (CASE) tools (Henderson-Sellers et al, 2000).

SSM focuses on human activity systems while UML use case models assume use of technology. The combined use of the SSM and UML-based methods yields a more balanced approach. Like SSM, Use Case Modelling is concerned with describing system behaviour. In SSM, the 'system' is that of human activity at a level of abstraction above implementation. Use Case Analysis was developed initially for people using computing systems; it can also be applied to the business process.

The Office of Government Commerce lists lack of effective engagement with stakeholders as one common cause of project failure (RAE and BCS Report, 2003). This research offers MoIST as one such means of assuring successful systems. MoIST offers three (3) development options that are able to fit several categories of information systems. This offers the developer and the client more flexibility and options in choosing the methodology most suited to the nature of their project. This should guarantee more success in project development. One of the major references on the quality of the systems development is the approach adopted. If the approach used is not appropriate for a particular type of application then it may limit the quality of the system being produced (Bennett et al, 2002). This means that there is a tendency to attempt to solve unstructured Information Systems problems by means of experimental and empirical research methods. The scientific method is still being used to try to find solutions to issues that are multi-faceted in their complexity (Bennett et al, 2002, p 57). There is therefore the need for a more relevant and unconventional approach. The alarming failure rate of Information Systems developments should at least have alerted Information Systems practitioners to the fact that hard systems engineering on its own does not seem to be working and applying the scientific paradigm to social organizational situations have failed (Checkland and Holwell, 1998). There must be an additional solution that can work in tandem with the hard systems engineering solutions to achieve increased success in a holistic manner.

MoIST has not been developed for every software development project. It is a method that can be maximally used for software projects where not much is known about the application domain, where the issues are somewhat unclear and systems thinking needs to be utilised.

Systems thinking of all the applied disciplines has demonstrated the greatest potential for linking theory and practice. Systems research is in advance of organisation theory in intervening in problem situations. It is ahead of operational research and management consultancy in its ability to think through implications at

the theoretical level and improve practice as a result. For applied disciplines, it can rectify any deficiency in their theoretical foundations. It also enhances its systems development and capabilities and enables systems to be built with the most important component in mind – the users.

Jackson, 1997

1.5 Research Overview

1.5.1 Development Concerns

- there is a need for effective means of integrating systems thinking into information systems design

- premature design decisions are being taken in the software development process before major requirements are known and opportunities for improvement have been agreed.

- An alternate means of more fully exploring the relationship between SSM, and existing information systems design techniques is needed.

- Options for software developers who wish to apply SSM and information systems design techniques in the context of UML are limited.

- There is a plethora of methods that involve detailed planning too early on in the software process which sometimes results in software that is difficult to change.

- There are many system design artefacts that clarify how the system retrieves and processes data. These however do not usually stipulate how to capture human activities and business processes.

- Stakeholder discussions based on class and sequence diagrams as communication tools are difficult to understand for clients not familiar with UML notation.

- Human factors are not appropriately integrated and factored into some methodologies used in the existing software development process.

1.5.2 Goal of research project

To design a method that integrates systems thinking, namely SSM into information systems design, using UML notation.

The application of Soft Systems Methodology integrated with a Unified Modelling language-based method is useful in software development for separating the ***'what'*** in what changes are needed?, from the ***'how'*** in how can requirements be met using information systems? This results in the requirements elicitation process being more accurate and efficient thereby increasing the success rate of software development projects.

- To redress premature consideration of system structure by using SSM within information systems design to identify and clarify the purpose of the system and the tasks needed for the achievement of those purposes.

- To develop a method that incorporates the results of SSM analysis into requirements specifications based on the UML.

- To encourage well structured and coherent debate about complex situations of software projects in order to decide on improvement strategies

- To emphasize designing systems supported by the information system instead of wrongly prioritizing designing the information system.

- To incorporate the human factor into the software development process.

1.5.2.1 Research Question 1

Is there a link between SSM and UML given their inherently different natures?

1.5.2.2 Research Question 2

Given that there is a link, how beneficial is that link to the software development process. In other words, what value does the link add to the existing software process?

1.5.3 Research contributions to existing work

This research has made significant and original contributions to the particular field of learning within which the thesis subject falls. Of these contributions some are general while some are local in scope.

General research contributions of the MoIST

- A method that explains the value to the software development process of a viable linkage between SSM and UML.

- A method that endeavours to more appropriately integrate the human factor into the existing software development process.

- An alternative method that provides a more wholistic exploration of the relationship between SSM and existing information systems design techniques.

- A method that integrates systems thinking into information systems design.

- A method that stipulates how to capture human activities and business processes instead of merely clarifying how the system retrieves and processes data.

- A method that minimizes the level of detail of the planning in the initial software process in order to achieve software that can be more easily changed along the software development cycle or path.

- A method that incorporates the results of SSM analysis into requirements specification based on the UML.

- A method that emphasizes designing systems **supported** by the information system instead of wrongly prioritizing designing the information system.

- A method that encourages well structured and coherent debate about complex situations inherent in software projects in order to decide on improvement strategies.

- A method that incorporates the human factor in a greater measure into the software development process.

- An alternative and additional method that incorporates the results of SSM analysis into requirements specifications based on the UML.

Local research contributions of the MoIST

- A novel means has been developed to integrate requirements analysis and design techniques from SSM into information systems design using the MoIST method

- An improved user requirement definition for certain types of software development projects

- A major intervention in the Academic Support Process in the School of Computing and Engineering using the MoIST method

- An electronic system 'ACcSys' that allows 'at risk' students to be quickly identified and assigned to the relevant personnel and resources.

- Design of ACcSys instruction documentation for Pathway and Module Leaders

- Development of a solution to the retention problem that resulted in redundancies in the School of Computing and Engineering

- Seamless merging of a developed electronic system as an interface to an existing electronic system within the university at minimal or no extra cost.

- An electronic system that is portable and is capable of being easily and successfully used by other departments within the university.

- An electronic system that can be utilised to solve problems of a similar nature in other universities.

- A system that can be used to facilitate the Personal Development Planning (PDP) mandate of the university.

- Creative work, namely a designed and implemented electronic system which forms a significant part of the intellectual enquiry. This original creative work was undertaken and invented as part of the registered research programme. This creative work is clearly presented in relation to the thesis and is set in its relevant theoretical and design context.

1.6 Conclusion

The thesis describes the problem to be solved and explains the methodology for solving it (see chapter 5). Related work is identified and discussed to show how the problem has been addressed before (see chapter 4). The shortcomings of existing work in the area are highlighted (see chapter 5). MoIST is then explained in terms of how it differs from other approaches and methods (see chapter 5).

The thesis supports the validity of the stated claim is valid through a presentation of references to prior work within both the soft systems thinking field and the hard systems engineering field. It further shows how previous work has influenced the new claim. Action research is then used to provide support for the claim. This is done by taking real life complex situations, applying the MoIST method and providing a major intervention and immense benefits at minimal cost. This action research is sufficiently documented, so that it may be reproduced by interested practitioners in related areas. Insight is then given into how the claim can be further used or extended in future work.

Traditionally software development has always focussed on deriving a definition of information system requirements. This usually results in information systems which are technically sound, but usually do not provide a large measure of satisfaction to the humans who use them (Mumford, E, 1995, 2003).

The methodology for this thesis makes a claim. This is that hard systems engineering by itself cannot claim to be the sum total of software development. The research states an important claim for an extension in the cultural vocabulary of technology. Here the cultural vocabulary of technology refers to the way in which technology

dictates a subset of usages, methods and approaches that have no bearing on or relation to human concerns (Maeda, 2002). This thesis asserts and justifies throughout its pages that only by 'enculturing' technology can we see an improvement in the success rate of software development products. Instead of concentrating on the technology alone, it is more beneficial to evaluate the interrelationships between the capabilities of the technology, the tasks being performed and the users of the technology (Maeda, 2002).

Chapter 2 - Soft Systems Methodology (SSM)

Thousands of engineers can design bridges, calculate strains and stresses, and draw up specifications for machines, but the great engineer is the man who can tell whether the bridge or the machine should be built at all; asking why, where it should be built and when.

Eugene G Grace, Former Chairman – Bethlehem Steel Corp (1916-1957)

2.1. Introduction

In this information age, clients' increased knowledge base has led to greater expectations from software developers. The software industry still tries to meet clients' demands. Consequently there have been many instances of software failures. Software faults may be traced to incorrect requirements. This is where the software matches the requirements but the behaviour recommended in the requirements specifications document is not appropriate (Pullum, 2001). There is a need to establish a stronger link between the requirements analysis and the design phases of the software development life cycle. Soft Systems Methodology (SSM) has been identified as one means of achieving that stronger link.

Soft Systems Methodology (SSM) was pioneered by Dr. Peter Checkland at the Lancaster University Management School, United Kingdom. SSM seeks to represent unstructured situations with the primary goal being to understand the situation as it really is. After understanding is gained, the methodologist or the owner is then empowered to make an intervention. This should result in some improvement to the previous situation.

'In [SSM] a number of notional systems of purposeful activity which might be 'relevant' to the problem situation are defined, modelled and compared with the perceived problem situation in order to articulate a debate …..''. SSM facilitates learning about an environment. It is not primarily geared towards achieving objectives. While not discounting the importance of achievement of objectives; SSM sets a chain of enquiry into motion in order to better perceive clearly, the nuances of a complex situation. As learning occurs over time, purposeful action may then be taken to improve the situation. (Checkland and Scholes, 1990, p.18).

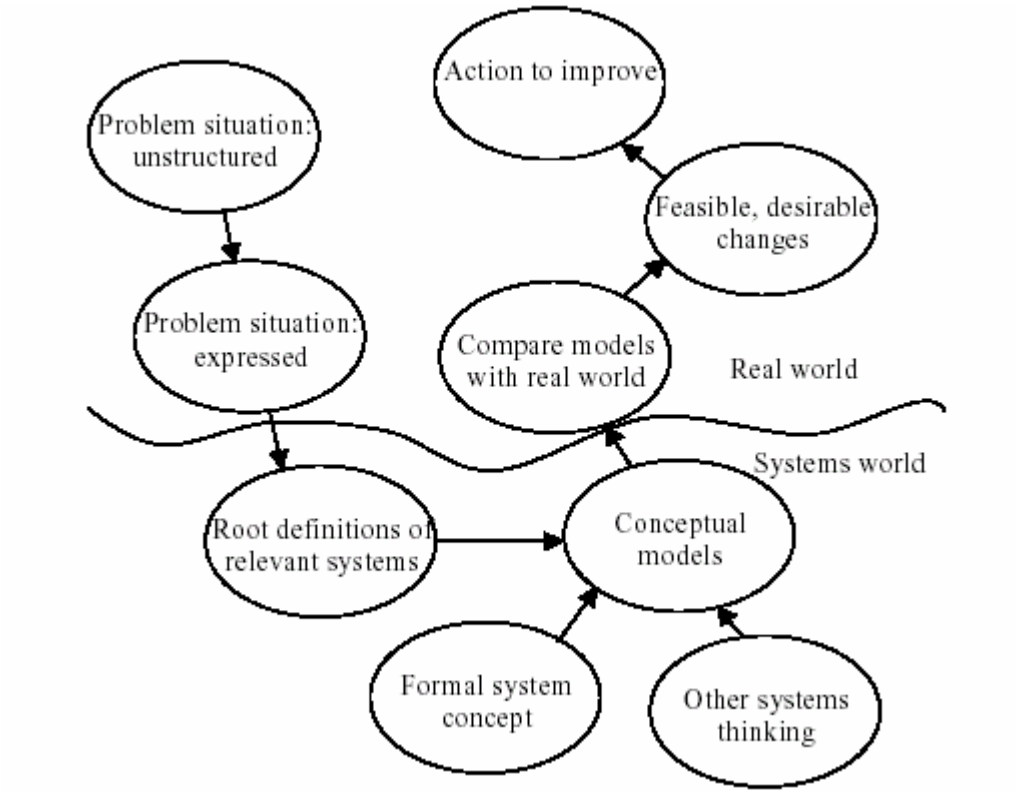


Fig 2.1: Original 7 stages of SSM devised by Checkland, 1981

2.2 Justification for the importance of SSM in this research

SSM is utilised as a means for resolving problems. It is used to extend the scope of feasibility study activities beyond those supported by hard systems engineering. SSM aids understanding of the many simultaneous views of an organization's goals. This facilitates potential interfaces to a system. It also allows factors affecting system implementation to be comprehensively investigated (see figs 2.1 and 2.2).

Traditionally software engineers tend to use class diagrams and UML artefacts such as use cases in the requirements elicitation stage to try to explain their understanding of the problem to the client users and to get feedback. This is usually unproductive as experience has shown that the technical level of these artefacts serves as a barrier and deterrent to user understanding. Systems engineering is excellent at dealing with technically defined situations. It however fails dismally in coping with the complexities of human affairs including management situations (Checkland & Scholes, 1999).

The soft systems approach provides the means for the analyst to discuss the situation in the users' own terms (CCTA, 1993). In Soft Systems Methodology (SSM) it is advocated that an information system should not be developed until the utilising system has been modelled. In other words it is unwise to build an information system until you have modelled the system that uses it. Soft Systems Methodology is a cyclic learning system which uses models of human activity systems to explore with the actors in a real world problem situation and their readiness to decide upon purposeful action which accommodates different actors' perceptions, judgements and values. Additionally, SSM is a methodology used to make sense of unstructured situations in real world organizations. (Checkland 1981, p. 98).

SSM is in essence action research. "This means that an action is taken and then evaluated. Action researchers contend that a complex social process can be studied best by introducing changes into that process and observing the effects of these changes." The approach used by organizational consultants must also introduce change, but in this case, the theoretical development and the rigorous empirical foundation are prerequisite elements of the activity (Baskerville, 1999, p25). This involves some action being taken. This triggers a resultant consequence that leads to learning of some sort occurring. This learning can then be applied to similar situations to increase the eventual learning.

It is sometimes not possible to define requirements accurately ahead of time. This is because the situation is new or the system being employed is highly innovative. The utilising system or the information system environment may change in reaction to the system being developed, thus initiating a changed set of requirements (Checkland and Scholes, 1990). In SSM, one is dealing with human beings. One therefore does not only need observation or design of human activity. What one primarily needs is decoding of this human activity. This answers the questions of *what* does it mean? And *what* does human behaviour mean? SSM also assumes that human beings should be free to choose their own futures. The ethos of SSM says that SSM is supposed to help by encouraging debate. There is no trick formula to working out what people mean. They have to be first engaged in debate (Checkland and Holwell, 1998).

The classical software development life cycle follows the progression of analysis, design, testing, coding. This cycle has been generally successful, but is limited in that systems are being developed that are not necessarily relevant to the needs of the clients. These systems are good systems, but not relevant systems (Checkland, 1981). This research applies SSM techniques to determine what the relevant systems are, before any development starts. This ensures a higher probability that clients are given systems that they need. SSM techniques look at the political and cultural ethos of an organisation and generate conceptual models of how the organisation really functions. This helps to determine the relevant systems; before one even gets to the analysis stage. The software development cycle then progresses as per usual to design and testing/coding stages.

The evolution of the soft systems methodology is somewhat independent of computer customers. It is a general problem structuring approach that may be used as a precursor to traditional information systems development. As the changing needs of computer customers and organisations have required more involvement in the development process; this has caused the SSM to be increasingly applied to information systems development. SSM is a reaction against the hard system view. This hard system view stipulates that anyone can see or engineer a hard system and perceive its minuses, faults and plusses. It is powered by observation and construction. This view does not always hold true as organizations increase in complexity and unstructuredness .

What is SSM?

SSM is a cyclic learning system which uses models of human activity systems to explore with the actors in a real-world problem situation their perceptions of that situation and their readiness to decide upon purposeful action which accommodates different actors' perceptions, judgements and values (Checkland 1984, p 98)

Checkland initially tried to use systems engineering as the framework for addressing ill-structured problems. This did not work very well as the systems engineering paradigm is different from systems thinking. Systems Engineering is a *'how*-oriented' activity; it answers the question. *How* can this need be met? *'What'* the need is has already been defined (Checkland & Scholes, 1990, p 17). This difficulty in using systems engineering gave rise to SSM. The whole core of SSM revolves around the following. There is a problem situation in the 'real-world'. An analysis then carried out of what exists at present in the problem situation. This is done by using a rich

picture, spray diagram or mindmaps, whatever depicts and captures the situation in the most accurate and understandable way. A definition of the relevant systems then follows. Here at this stage a root definition, RD of a system relevant to the problem is created.

It is highly recommended that there should be several RDs. In creating or constructing the RDs , it was made clear that at stage 5 [comparison between real world and the conceptual model], it is difficult to avoid seeing the RD and its model as normative, if there is only one definition and one model (Checkland, 1981 p. 208). The way to avoid this is to entertain several root definitions, best of all including incompatible ones and to make models based on more than one of them. The next stage is a conceptualisation of 'formal systems' to carry out the functions described in the 'root definition'. This means that the root definition is here subsequently modelled by a set of minimum necessary activities shaped in terms of the 'formal systems or conceptual model'. Comparison between the rich picture representation and the conceptual model follows. There is then a definition of a range of possible changes that could positively intervene in the real world problem. A selection of a desired, agreed-to-be-feasible change. Design of the agreed change follows and ultimately implementation of the agreed change.

Phases of SSM

SSM in its most basic form has four (4) distinct stages. The first involves the existence of a real-world situation of concern. Someone is concerned enough to initiate a finding out or an investigation stage. The first stage is therefore the finding out stage and the artefact produced at this stage is usually the rich picture.

The second involves constructing models of relevant human activity systems (HAS) or holons. These HAS or holons are then named and modelled and should not bear too much resemblance to the real world situation identified in the first stage. With SSM, firstly a rich picture of the current environment is drawn. This rich picture is uncoloured or unbiased by systems terminologies, secondly, ways of using systems ideas in problem situations are then developed based on that view. Thirdly, there is modification of both the systems outlook and the ways of using the systems ideas as experience is gained, as mistakes are made and as lessons are learned. There is then reflection on the interactions between systems thinking and systems practice in order to draw conclusions which will allow future theory to benefit from practice and future practice from theory. There are four techniques usually used in this stage. The Root definition (RD), the CATWOE, PQR (see below) and conceptual models (CM) of

Human Activity Systems (HAS). The root definition is a brief formulated statement that best describes the system and tells what the system will or should do. CATWOE is a mnemonic used to ensure the well-formedness of each root definition.

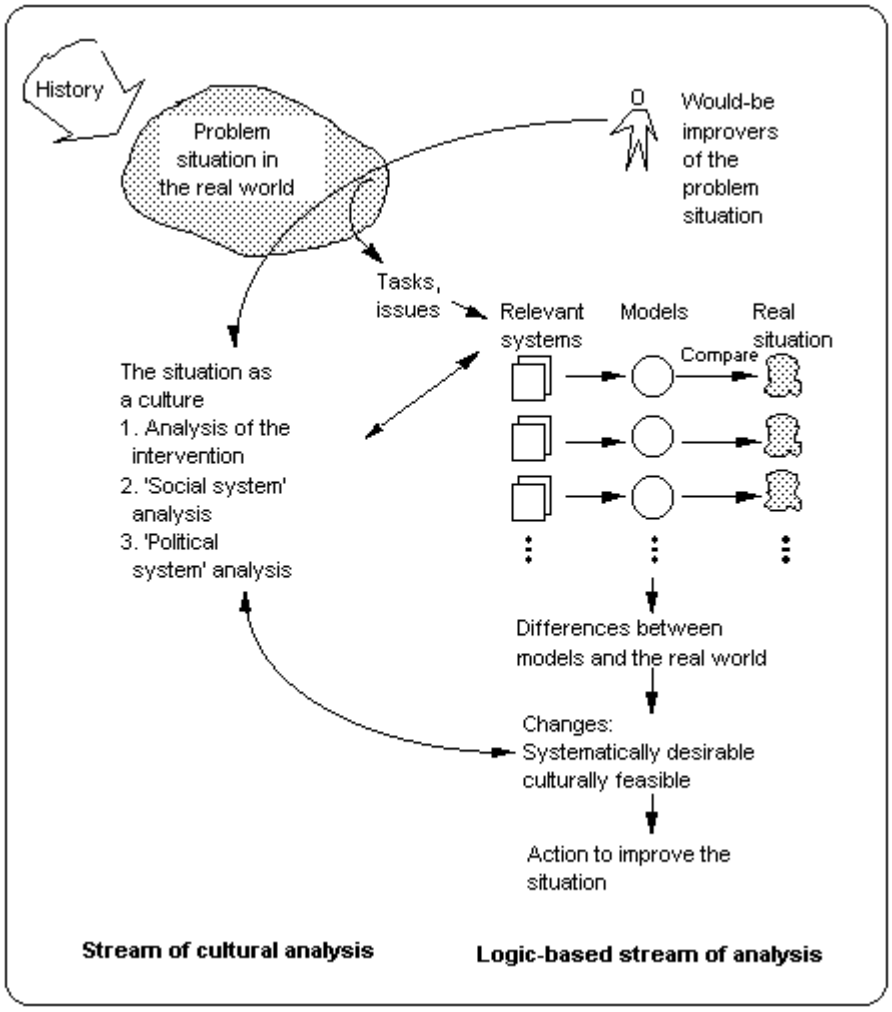


Fig 2.3 Social, political & cultural analysis, Checkland et al, 1998

Phase 1 – Rich Picture

It is possible to begin the process at any phase. However it is the relationship connection between the phases as opposed to their order that is crucial. SSM usually begins in phase 1. Here an exploration of a real-world situation of concern is initiated. This is usually because someone perceives a situation as problematic and desires to fix it in some way. The purpose of this exploration is usually to provide a more accurate comprehension of the situation. It is also to identify pertinent issues that need to be addressed. The Phase 1 findings are usually summarised in a rich picture, Checkland and Scholes, 1990.

The process of gathering information, appreciating the situation and then identifying relevant issues adhered to certain guidelines. These were to look for 'elements of slow to change structure' and 'elements of continuously changing process'. These elements have in later versions of SSM been developed and replaced with the guideline to explore the situation through analysis of the intervention – Analysis One, social system analysis – Analysis Two and political system analysis – Analysis Three, Checkland and Scholes, 1990, p 47).

Phase 2 – Conceptual Models of Human Activity Systems

From the rich picture, relevant issues for improving the problem situation are selected and modelled. These relevant models of purposeful activity are intellectual devices used to stimulate and structure the debate about the situation being studied. They are also used to focus on concepts of pure purposeful activity from a certain perspective Kareborn, 2000). That is why they are referred to as conceptual models of 'human activity systems', Checkland and Scholes, 1999). Human activity systems comprise all activities that are carried out by human beings.

In order to form a whole, these activities are linked by some principle of coherence or some unifying purpose or mission. These conceptual models should not be accounts of the real world, or Utopian designs, but should be considered as epistemological devices that help to structure a debate, Kareborn, 2000).

Phase 3 – Comparison

In this phase, the human activity systems are compared with actual perceptions of the situation. Through the comparison and the debate it creates, fresh insights are gleaned. This allows for accommodations between different interests and perspectives. These accommodations must be both feasible and desirable. They usually result in actions that can improve the situation.

Phase 4 – Action to improve the situation

These actions to improve the situation are the start of Phase 4. After the implementation of an agreed change to improve a situation, the original problem situation is either resolved or is transformed into a new situation of concern. If changes cannot be agreed upon, a more extended examination of relevant systems might be needed.

The four (4) phases described above represent the systems thinking part or the modelling phase of SSM. The main function of the modelling phase is to highlight different perspectives of the problem situation and to structure the thinking about the situation. To achieve this, four (4) precise techniques have been developed. These include Root Definition (RD), CATWOE, PQR and Conceptual Models of Human Activity Systems (HAS).

SSM techniques

1. Root Definition

Root definition means naming a system of purposeful activity in a succinct statement. The official guidelines for a well-formulated root definition is that it should contain the elements of the mnemonic word CATWOE or PQR, Checkland, 1999).

2. PQR

PQR refers to the statement 'Do P by Q in order to contribute to the achievement of R'. PQR also answers the three questions: What to do (P); How to do it (Q); and Why do it (R)? (Checkland, 1999).

3. CATWOE stands for:

Customers -	the beneficiaries or victims of the system;
Actors -	persons who perform the transformation process;
Transformation -	an input-output process by which some entity is changed to a new form of that same entity;
Weltanschaung -	a worldview which makes the transformation meaningful;
Owners -	the persons who can stop the transformation;
Environmental constraints -	elements which affect the system, but which cannot be controlled.

2.6 Case Study : A Study of the learning and teaching strategy delivered to Business Computing Students: Towards determining the most effective approach

The case study below is used to illustrate an example of how SSM is actually used in an unstructured, complex, real life organizational situation. It demonstrates the typical SSM stages and was conducted at the School of Computing and Engineering.

2.6.1 Introduction

Students of differing faces and places contend with a multiplicity of issues at various times. In recent times, the challenges they face seem to have intensified and they appear to need more effective teaching and learning solutions tailored to their specific needs. This study seeks to explore and expose the issues and then to make valuable recommendations. The academic programme in most universities is divided into manageable units called modules. The learning and teaching strategy is usually based around the construction and delivery of these module entities. A module may be defined as a related logical sequence of instructions that is executed by a facilitator for the benefit of learners with the aim that the learners should totally master instructions, be able to replicate them and apply them in novel situations.

People learn differently and teaching has to be deliberately structured in order to get the desired learning outcomes. All students are individuals who assimilate new information in different ways. One student may feel more comfortable reading information, another may be more practical in orientation, yet another may feel perfectly at ease doing both. Teachers need to determine students' expectations and experiences and tailor subsequent classes to take these into consideration. This helps to manage the learning and teaching process in such a way that the student is not disadvantaged. Some teachers believe that learning is the student's responsibility while others believe that they are responsible to ensure student learning. Whatever camp the teacher falls into, there must be ways of ensuring that the student learning experience is maximized.

2.6.2 Background

Traditionally students in Higher Education followed the Oxbridge model. Stereotypically they were intellectually advanced and financially secure. Consequently there was not much need for them to have part-time jobs.

In the last ten years a new specie of students has emerged from the ashes and difficulties associated with Higher Education. These are 'commuter students'. They can be further classified as semi-distance learners. By definition, these students stand in stark contrast to the traditional Oxbridge model. They live at home with their parents and not on halls of residence. They average one hour of daily commute to get to classes. Higher Education now comes at a greater cost so they work at part-time jobs while being full-time students. Most are privy to computer access at home. The phenomenon of new universities has birthed these 'commuter students'. The government's mandate of Higher Education and widening participation has ensured that more students get access to a university education, but at a cost to the education system and to the students themselves. With the rise of the 'commuter student' has come the fall of retention rates when compared with rates of past years, retention levels are still falling at an appalling rate. The 'recruit more' and 'retain more' goals are now increasingly being perceived as mutually exclusive. Therein lies the challenge. There is therefore the need to retain a greater number of students without compromising the academic standards. There needs to be a way of enabling learning and delivering teaching to new millennial students with all the attendant problems. The pressing demand is for high retention levels to be maintained without diminishing academic rigour.

HOW THE PASS RATE HAS CHANGED

Top 10 subjects Overall Pass Rate

	Pupils	1993	2003
English	78,746	84.3	98.4
General Studies	58,430	72	90
Maths	55,917	82.4	94.4
Biology	51,716	77.4	92.6
History	42,018	83	97.7
Psychology	41,949	76.8	94
Art and Design	38,314	90.5	96.6
Chemistry	36,110	79.9	95.1
Geography	35,749	81	97.9
Business Studies	33,133	80.1	96.4

Table 2.1: Pass Rates cited from a daily newspaper – Friday, August 15, 2003

There is another type of student in the education arena. This student relishes the opportunity for distance learning education. This is fast becoming ubiquitous and distance education is a thing of the present and future. Yet another category of student is the experienced professional who for a variety of reasons, did not pursue formal university education. Pressed with corporate, family and personal responsibilities, the desire and dream for Higher Education seem a distant dream; but the hope for attaining that dream still burns brightly.

One of the six key objectives of a learning and teaching strategy is 'to support the continuing development of high quality, responsive teaching, learning and assessment that recognises the diversity of learners, the appropriate use of learning technologies and the dispersed nature of the university (Dearing report, Higher Education in the Learning Society NCIHE, 1997).

These are only three (3) of several categories of learners whom the learning and teaching strategy have to be configured for. The commuter students who now have to fund themselves, the students for whom distance education is more beneficial and the working executives who desire to pursue Higher Education; but are currently constrained by other factors. This study uses SSM to explore the issues involved and make recommendations as to the best way forward.

Methodology of the Research

Soft Systems Methodology (SSM) was the vehicle used to facilitate this study. SSM was pioneered by Professor Peter Checkland at Lancaster University. SSM allows for the study of an environment in order to extract pertinent factors from that environment so learning of the environment can occur. This learning then can precipitate relevant change in an organisational environment.

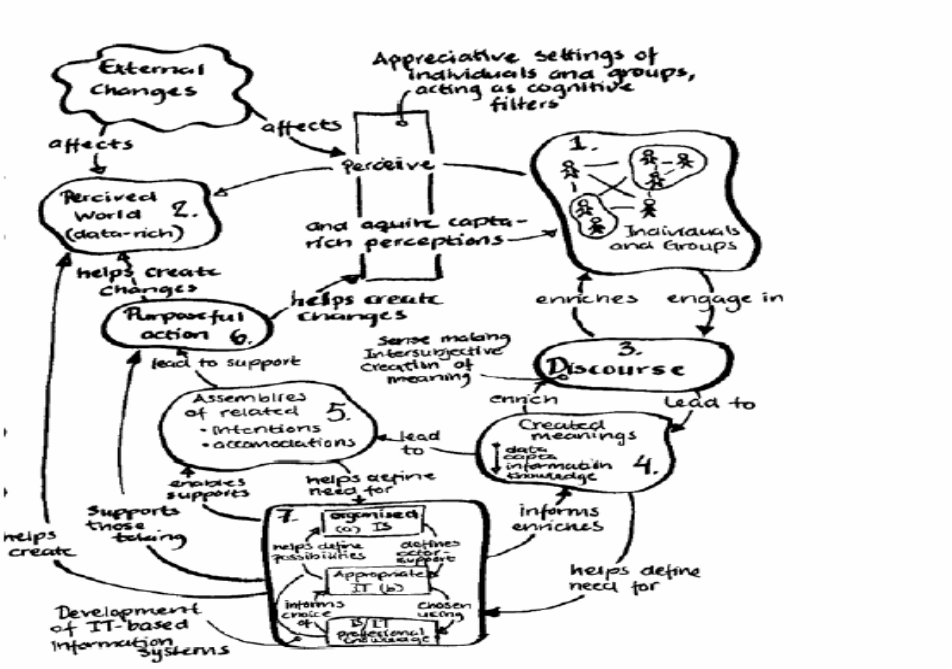


Fig 2.4: POM Model of SSM

The POM model within SSM is comprised of seven (7) elements. It was developed to try to make sense of the information systems field, Checkland and Holwell, 1998. Element one consists of the people as individuals and group members. Element two is a data rich world perceived through assumptions. Element three is where meaning is created inter-subjectively. This feeds into Element four which embodies the attributes of meanings based on information and knowledge. Element five accommodates conflicting interests and element six is the purposeful action taken. Element seven represents information systems that support organisational members. The POM model represents organisational information and communication processes, sense making and processes of creating shared meaning in a never ending learning process (Holst et al, 2004).

The finding out process is vital and integral to SSM. This is where the client's desires are captured from the methodologist's perspective. When a complex situation in an organization is presented, the person doing the investigations brings to bear on the finding out certain competencies and skills which are personal and unique. These competencies and skills influence the quality of the output of the investigation. In order to better appreciate the problem context and achieve better information gathering; the methodologist should pay attention to 'elements of slow to change structure' and 'elements of continuously changing process (Checkland, 1981). He

further suggested that one should see how the elements all worked together in the situation climate. In however, this suggestion was revamped in favour of Analyses One, Two and Three (Checkland and Scholes, 1990). SSM provides a flexibility in the information gathering devices. Some methodologists find it easier to work with rich pictures, some with mind maps.

2.7 Analyses 1, 2 and 3

SSM has innovated another approach to the "finding out" process in a problem situation. This approach uses three related analyses. The first is referred to as Analysis One. This is the Analysis of the intervention. This looks at the existing situation and establishes who in the situation is in the role of 'client'. The client is the one who causes the intervention to take place. The role of the would-be-problem – solver is also established. This is the person who conducts the study. This person compiles a list of potential problem owners. This activity is related to the concept of stakeholder analysis in requirements engineering (Sharp et al, 1999, Smith, 2000).

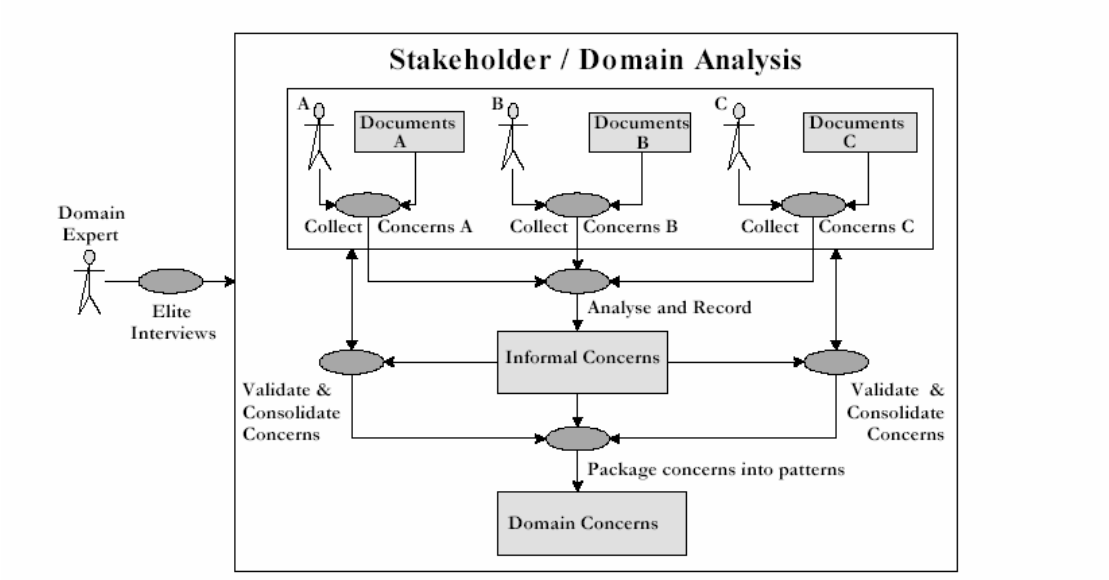


Figure 2.5: Stakeholder/Domain Analysis, Sarkar, P, 2002

Analysis Two examines the cultural aspect of the situation. It sees the situation as a social system and seeks to discover the social roles present in the situation, the norms of behaviour that are expected to be displayed by those in the social roles and establishes the values that determines the measure of whether or not a role holder's performance is considered good or bad.

Analysis Three addresses the political aspect of the situation. It examines the disposition and the nature of power and authority displayed in the situation. It looks at what commodities of power are present in the situation. It also seeks to identify how the commodities are obtained, used, preserved and passed on.

Analyses used in the 'Finding out stage'

Analysis of the Intervention – Analysis 1
• Who is the Client? • Who is the problem solver? • Who are the problem-owners?
Social System Analysis – Analysis 2
• Role Analysis • Norms Analysis • Values Analysis
Political System Analysis – Analysis 3
• Identifying possible conflicting interests and personality clashes • Interests resolvable by accommodation • Differences of interest resolved by use of power

Table 2.2: Analyses 1, 2 and 3

<u>Case Study demonstration of Analyses 1, 2 and 3</u>

Analysis 1 – Analysis of the Intervention

Client – Module Leader
Problem-Solver – Soft System Methodologist
Problem-Owners – School of Computing, Module Leader

Analysis 2 – Social System Analysis

Roles	Behavioural Norms	Values
Student	Aims to maximise their potential by gaining the knowledge, attitudes and skills needed to meet the requirements of work and society	Attending required classes and tutorials **GOOD** Getting assistance when needed **GOOD** Dropping out after the first semester **BAD**
Lecturer	Teach students in such a way that learning takes place and the goals of the module syllabus are met	Use innovative ways of teaching to enhance student learning **GOOD** Provides and point to high quality support resources for student **GOOD** Disseminate course content without adequate explanation **BAD**
Module Leader	Coordinate all the resources and personnel relevant to a module to ensure that it functions smoothly	Liaise with support tutor And provide necessary materials in advance **GOOD** Examines and amends module to consistently upgrade module quality **GOOD** Disregards students concerns and feedback **BAD**
Undergraduate pathway leader	Oversees the smooth running of all modules in the undergraduate pathway	Communicates effectively with all module leaders **GOOD** Has no future plans to improve and upgrade current pathway **BAD**
MSc pathway leader	Facilitates the efficient running of the MSc pathway	Knows what the current industry needs are for the pathway **GOOD** Has no idea of how the modules are being taught

		BAD
Distance learning coordinator	Provides innovative ways of student learning through the use of distance learning materials	Has experience in how distance education works in practice **GOOD** Installing outdated modes of distance learning **BAD**

Table 2.3: Analysis 2

Analysis 3 – Political System Analysis

Disposition of power

The stakeholders detailed above represent different parts of the learning and teaching spectrum. Consequently there is no stringent or ordered hierarchical structure represented there. One thing they all have in common is authority and influence over the student. In both the undergraduate and MSc streams, a pathway leader oversees a module leader. Both functions incorporate the lecturer role. Any conflict or communication breakdown would be more likely to happen between those two roles. The distance learning coordinator is an autonomous position and usually indirectly relates to the other lecturers' responsibilities.

Nature of Power

The power reflected here is the power to improve or retard the learning and teaching process in the school. The manner of execution of the responsibilities is paramount in ensuring that student learning is enhanced and effective which ultimately achieves the corporate mission of the university. Any improprieties in carrying out the functions above could result in even lower retention rates and recruitment levels of students as the bad news spreads.

Rich Pictures

Rich pictures are used in SSM to capture the essence of the situation under study. Political, social and cultural realities are captured in a pictorial or diagrammatic form. This enables the methodologist and the stakeholders to see the whole picture at a glance. It enables them to understand the various forces at work in their particular situation and see it through another's eyes. This increases their objectivity and helps them to more accurately see with the SSM methodologist what the next step needs to be. The rich picture usually represents the methodologist's perspective on the problem and gives their viewpoint. The thought processes required to develop the rich picture give rise to deep understanding of the situation and provoke thoughts about relevant human activity systems which lead to conceptual model building (CCTA, 1993, p36). One does not need to be a Picasso or Michelango to be able to give an accurate depiction of happenings and events and relationships in a project situation. Stick persons and cartoon like characters will suffice. SSM encourages the analyst to broaden the initial concern of study away from the detail of the intended information system and towards the situation in which the information system is expected to provide useful support. The first task is to describe the situation (CCTA, 1993, p34).

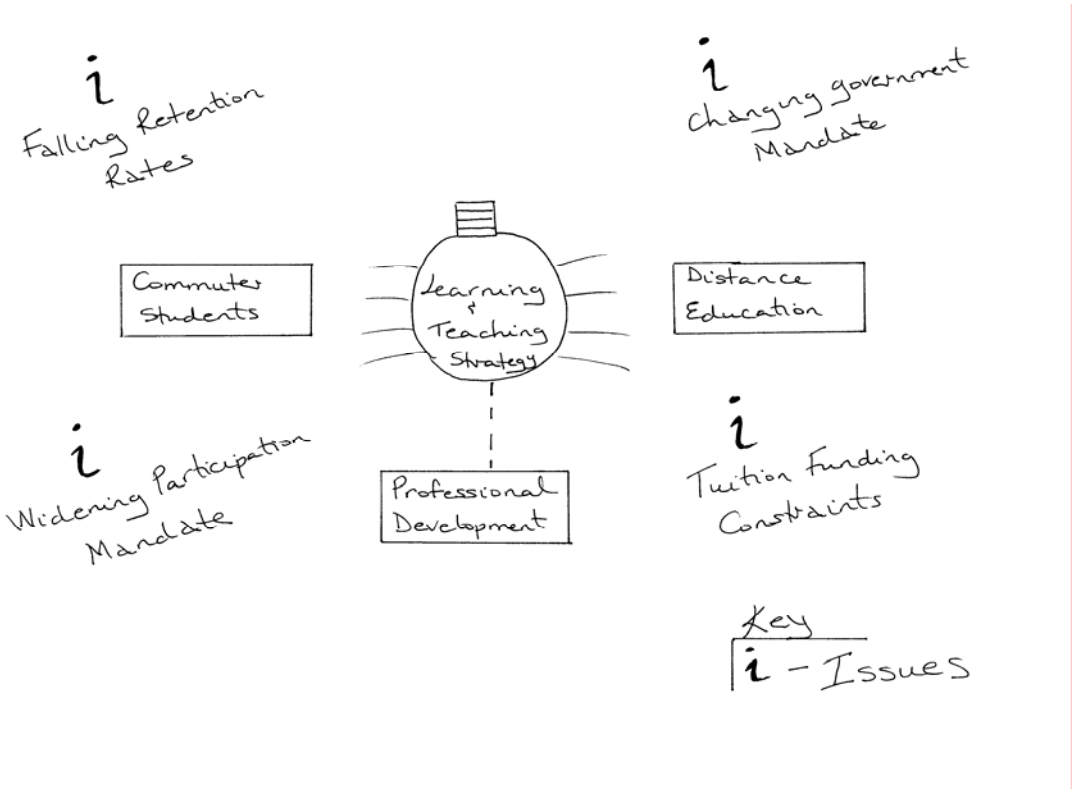


Fig 2.6: Rich Picture of Teaching & Learning Process (created by Hopkins, 2004)

The process of creating rich pictures is not very standardised. There is no one set way to draw or terminology or jargon for this. The depth of its richness is usually left up to the methodologist. The process of creating rich pictures also serves to tease out the concerns of the people in the problem situation. These soft facts include the sorts of things that the people involved in the situation are worried about, conflicts, politics and other concerns (Avison et al, 1990, p45). It has been the experience of this researcher that the initial effort of starting to draw a rich picture was somewhat challenging. The question was always where to start. The challenge is greater for those methodologists who do not perceive themselves to be artistically inclined. Relief and comfort came from several directions. The first was the realisation that there is no one correct rich picture. It is subject to the methodologist's interpretation and therein lies its acceptance. Research has shown that anyone with a 'normal' brain ie (not genetically or physically damaged) can learn to draw to good art school level. The reason so many people assume they are incapable of creating images is that, instead of understanding that the brain always succeeds through continued experimentation, they mistake initial failure for fundamental incapacity and as its true measure of their talent. They therefore leave to wither and die a mental skill which could have flourished naturally, (Buzan, 2000).

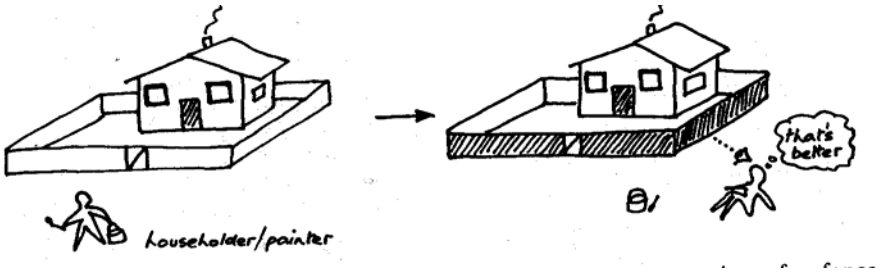


Figure 2.7: Rich Picture, Checkland, 1990

How to start drawing rich pictures

1. write down all the known stakeholders
2. think about who interacts with whom or what
3. start by drawing the main interaction in the centre of the page
4. use a key to denote the meaning of symbols

Selecting Relevant Systems

Relevant systems in SSM are called root definitions (RDs). As one expert put it 'systems thought to be relevant to that deeper exploration of the problem situation which will lead to action to improve it'. Originally, it was thought very important to get the correct relevant system. The passage of time has however softened that view and now it is not thought to be incorrect to formulate as many relevant systems or root definitions as needed. The most relevant one can then be selected.

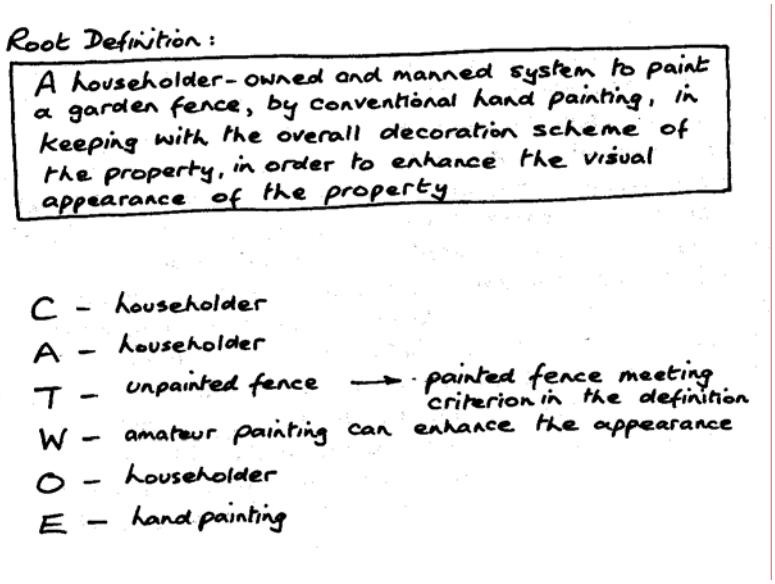


Figure 2.8: Root definition and CATWOE, Checkland, 1990

Root definitions used to be formulated without a specific standard or template. Work done by Smyth and Checkland in 1976 saw the birth of the CATWOE which was a mnemonic for the new standard by which all relevant systems should be measured and drawn up against. Relevant Systems are therefore constructed as root definitions done in accordance with the CATWOE. Relevant systems are usually divided into 'primary task definitions'. The most important part of a root definition is the transformation 'T'. Here some input is transformed or changed into some output. This poses one of the most challenging bits in the root definition formulation. The mistake that everyone usually makes is to state the resources needed for transformation as the input.

Case Study Example

'to provide innovative approaches to student learning by developing new teaching methods to complement those already in place, providing students with IT, information management and other study skills, enhancing student employability and career skills and by extending the learning opportunities in the workplace in order to provide a better match to the needs of the changing student profile, be more responsive to change and encourage increased participation in HE by students from non-traditional backgrounds.'

Table 2.4: Root Definition of Teaching and Learning

Formulation of root definitions is one of the main activities of SSM. A root definition is a precise description of the emergent properties of a system. A root definition should explicitly contain the CATWOE elements. These six (6) elements, Customers, Actors, Transformation, Weltanschauung, Owner and Environment are closely connected to the idea of human activity systems (HAS), (Mathiasen et al, 1994).

Conceptualising data

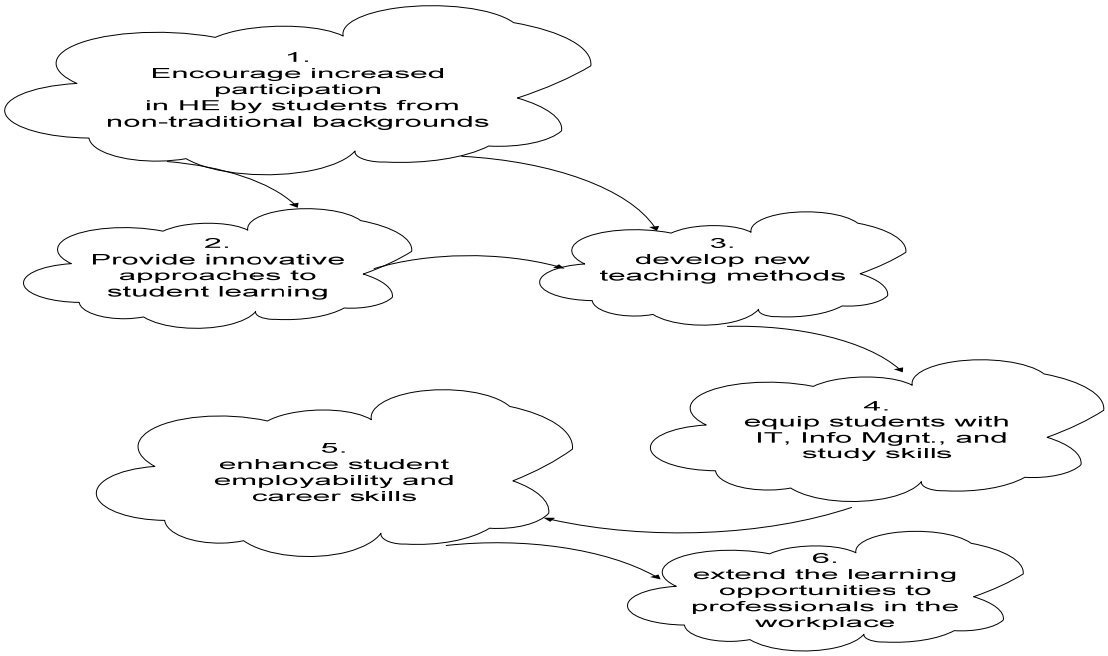


Fig 2.9: Conceptual Model of Teaching and Learning Process

Case Study illustration of Conceptual Model

IN SSM, a conceptual model contains the minimal set of related (human) activities needed to carry out the transformation described in the corresponding root definition. A system is thought of as being adaptive, therefore monitoring and controlling activities are built into each model. Subsequently, the conceptual model must be defensible against the root definition and the root definition against the conceptual model, (Mathiasen et al, 1994).

The conceptual models give life to the activities of the relevant systems expressed as the Root Definition. These activities of the systems are verbs that are assembled together according to logical dependencies. In other words, their linkage is provided by some cohesive factor or common purpose. These models should be neither accounts of the real world, nor utopian designs, but rather epistemological devices which help to structure a debate (Bergvall-Kareborn, 2002).

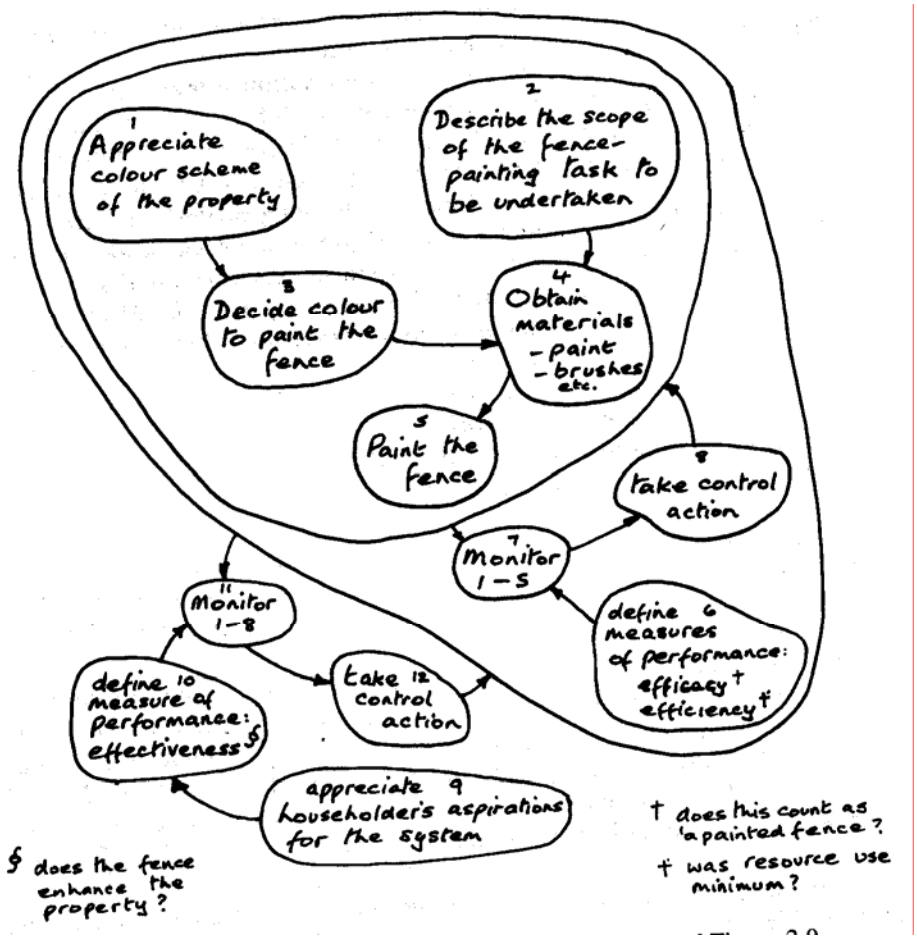


Fig 2.10: Conceptual Model, Checkland, 1990

Conceptual models built at stage four (4) of the methodology are neither descriptions of actual human activity systems nor accounts of such systems which ought to exist.

Their purpose is only to generate a high quality discussion with concerned participants in the problem situation, (Checkland, 1981, p 236)

Monitoring and Control

Monitoring and control in conceptual modelling seeks to find answers to the question, 'How could this system fail? This monitoring and control mechanism evaluates the activity system's performance against 3 and sometimes 5 measures of performance. These are known as the 3 or (5) Es

E1 Efficacy does the means work? Are the activities accomplishing the transformation T
E2 Efficiency Are minimum resources used; could the transformation be accomplished with less resources?
E3 Effectiveness Is the right thing being done; does the transformation help to attain the long-term goals related to the Owner's expectations
E4 Ethics Is the transformation a morally correct thing to do?
E5 Elegance Is the transformation aesthetically pleasing?

Table 2.5: Measures of Effectiveness – the 5 Es

Case Study illustration of CATWOE

C – Customer	Students
A - Actor	Distance Learning Coord, Pathway leader, Module leader
T - Transformation	ineffective teaching and learning strategy -> relevant and more effective teaching and learning methods and strategies
W – Weltanschauung	universities are faced with many external and internal challenges that have implications for their strategies for teaching, learning and assessment
O – Owner	school of computing, university
E – Environment	financial and suitable personnel resource constraints

Case Study Illustration of Comparison Phase

Conceptual	Reality
Provide innovative approaches to student learning	There is effort being expended by the university to provide innovative approaches to learning. The academic skills unit has been trying to help students overcome handicaps in various learning areas. There is also the personal tutor system
Develop new teaching methods	The Blackboard v 6 intranet is being utilised by lecturers and students to make learning and teaching more effective. Distance learning courses are being developed in the school by the coordinator
Equip students with IT, Info Mgnt. and study skills	There are programming support classes in the school and study skills support is available from the academic skills tutor
Enhance student employability and career skills	The status quo for this has not changed, but there are plans afoot to promote personal development planning (PDP) and encourage students to keep personal progress files. This is thought to increase employability at the end of the academic programme
Extend the learning opportunities to professionals in the workplace	Currently such opportunity is not formally in place. With the advent of the distance learning modules, this can change
Encourage increased participation in HE by students from non-traditional backgrounds	Widening participation of 50% of the populace has been mandated by the government. One effort is the running of an induction day called Quick Start for students with conditional offers from FE colleges to encourage their retention in the programme.

Table 2.6: Comparison Phase

Implementing Change

The models of human activity systems are compared with the actual situation. Consequently this comparison generates debates and fresh insights are revealed. This can then lead to satisfactory accommodations between the differing viewpoints. These accommodations must be systemically feasible and culturally desirable.

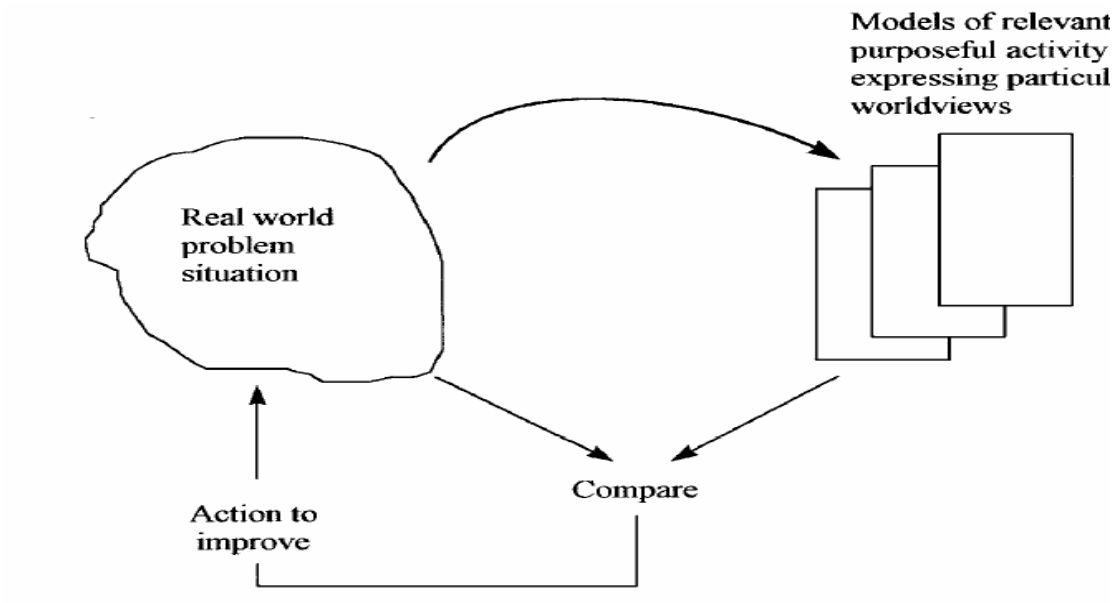


Figure 2.11: SSM as a Learning Cycle, Checkland and Scholes, 1990.

Defining change

This SSM study brought to light quite a number of issues. Not all of them will be addressed, given the scope of the project and the time factor of the research. The changes defined below were the ones thought to be implementable given the constraints.

Change 1

In order to be motivated to learn, students have to be interested in their work. One of the best ways to motivate is to provide a variety of activities. One way to do this is to utilise the existing Blackboard version 6 (BB6) in the School of Computing and Engineering. Blackboard has facilities for generating relevant subject quizzes and automated marking. This could be done after each unit is administered. It offers spreadsheet view of students' performances on these quizzes and the module leader can more effectively monitor how each student is grasping the material presented.

Change 2

The distance learning coordinator has developed a software that enables students to learn at their own pace without being physically present at the university campus. The proposed change is that the software can be used to develop one or more existing for the MSc programme. This will enable the core areas for business computing and information systems students to be made available in distance learning format

Change 3

To offer the above created distance module or modules to professionals in the workplace who are not able because of various constraints to physically attend classes at the university

2.8 Limitations of SSM

SSM with all its enumerated virtues above is not without its detractors. Some of its perceived limitations especially as it relates to information systems development are highlighted below.

- Critical Systems Thinking philosophy (CST) criticises SSM for its interpretivism. CST's critique towards SSM points to the lack of "objective" standards for the interpretations'. CST says that the critical results of SSM rests on persons and groups in the real world situations participating openly and in a shared spirit (Bergvall-Kareborn, B, 2002, p 474). The concern however is that this is not a perfect world and the balance of power in organizations is not equally distributed. This could corrupt the very method used to determine relevant systems as it is the results from the group participation that is used in the recommendations for software development. 'the kind of open, participative debate which is essential for the success of Soft systems approach, and is the only justification for the results obtained, is impossible to obtain in problem situations where there is fundamental conflict between interest groups which have access to unequal power resources (Jackson, 1991).

- From a learning perspective, it is a weakness that the rich picture is not consistent with the conceptual model. For example, the ethical and aesthetic criteria for measuring the performance of the conceptual models are nowhere to be found in the rich picture. Hence two new aspects of evaluation which were not considered in analyzing the problem situation are suddenly introduced to the conceptual model.

- SSM is interpretivistic and lacks 'objective' standards for the interpretation

- SSM has a limited domain of application, a regulative character, and a tendency to retain status quo owing to its subjectivist approach to social science, its interpretative assumptions, and because the approach did not attempt to ensure the conditions for 'genuine' debate

- The kind of open, participative debate which is essential for the success of the SSM approach, and is the only justification for the results obtained is impossible to obtain in problem situations where there is a fundamental conflict between interest groups which have access to unequal power resources. (Jackson, 1991, p 133)

- SSM is seen as subjective and pluralistic. This is its main strength. Ironically this main strength has been the characteristic of SSM that has been most criticised.

- SSM has a tendency to result in regulatory, rather than radical, agendas for change.

- The epistemological meaning underlying concepts and ideas within SSM may prove difficult to grasp for persons not familiar with the interpretative tradition.

2.9 Related Soft Methods

In order to redress the deficiencies of traditional organisational models, other process based modelling approaches have arisen in addition to SSM. These include Vickers appreciative system, Beer's Viable Systems model, Business Process Reengineering and Participative System Design.

2.9.1 Vickers Appreciative System

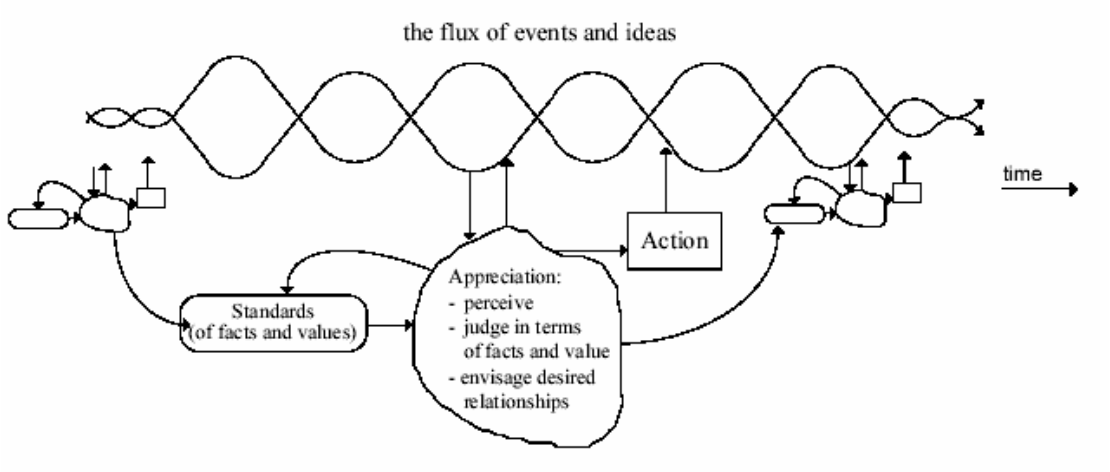


Fig 2.12: Appreciative and learning system, Checkland and Casar, 1986

In 1963 after retiring, Vickers had more time on his hand to try to gain a broader understanding of more than forty (40) years of professional experience by seeking an understanding of human affairs in general and organisational life in particular.

Vickers ideas were depicted diagrammatically by (Checkland and Casar, 1986). The model starts with Vickers two –stranded rope which is an interacting flux of events and ideas. This flux is interpreted, valued and judged according to standards created by previous experiences. Observing 'what is' and comparing it with the standard, is known as appreciation. Vickers aimed to understand social and organisational processes. He showed how information and communication processes lead to change of our appreciative settings, which leads to the basis of our decisions (Holst et al, 2004)

2.9.2 Beer's Viable Systems Model (VSM)

VSM demonstrates central processes and information flows to and from its management system. It shows functions necessary for an organisation to be viable. One criticism of VSM is that it insufficiently represents modern organisations due to its hierarchical structure. This results in the information flow becoming vertical and more controlling (Holst et al, 2004).

2.9.3 Business Process Reengineering (BPR)

BPR advocates radical change in organisational processes (Hammer and Champy, 1995). Proponents of BPR insist that implementing the proposed ideas will enact revolutionary organisational improvements. Detractors however say that the top down approach of BPR results in a high failure rate. It is also claimed that it downplays the role of people and knowledge within the organisation (Holst et al, 2004, Galliers, 1997).

2.9.4 Mumford's Participative Systems Design

One of the main aims of the ETHICS method is to achieve a better balance between technology and people. Traditionally, economic and technical objectives have dominated the thinking of those designing new systems. The specifications for new systems have tended to be more skewed towards technical efficiency objectives and very rarely with human needs and interests (Mumford, 1995).

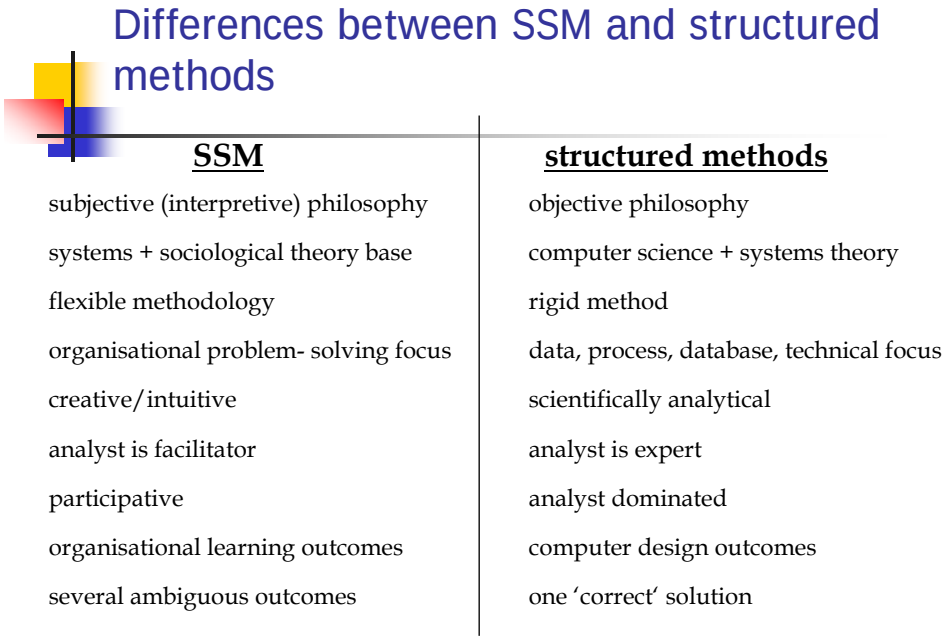


Table 2.7 - Table adapted from Rose, J, http://www.cs.auc.dk/~jeremy/resources

2.10 Software Development Method Building in General

In order to fully appreciate the merits and demerits of SSM, it is vital to examine other software method building work in general. This helps to put SSM, UML based, MoIST and all other methods in the research into their proper context. Examining general software methods also gives a more balanced perspective and understanding of each respective method. This is important for understanding the role MoIST and other methods play in the software development process as a whole.

A **method** is defined as a procedure, a systematic way of doing anything according to a regular plan or as the mode of a procedure of accomplishing something, (Grossett, 2000). A **methodology** on the other hand is the philosophical analysis of method and procedure or the method and procedure used by a science or discipline. There have been keen debates and some contentions as to the distinction between method and methodology. In this section, general software development methods are examined. A brief history of methods is presented and the varying methods, their contributions and their drawbacks are contrasted and compared.

Methodology may be defined as the logos of methods or principles of methods used to achieve a process. Another definition is a set of principles which have to be adapted in use to a particular situation. (Bennett et al , 2002, p 57) assert that 'a methodology consists of an approach to software development, a set of techniques and notations that support the approach, a life cycle model to structure the development process and a unifying set of procedures and philosophy'.

Appropriate use of methods in a software development project results in a more accurate project and a more balanced research. (Bennett et al 2002, p 556), stated that 'in practice, methodologies vary widely in philosophy, in completeness of definition or documentation, in coverage of the life cycle, and even in the type of application to which they are best suited'. The method and the methodology of any process, project or endeavour could be argued to be the most vital and crucial component to that endeavour. The methodology provides the framework of justification for the process.

A methodological framework exposes the similarities and differences between methods. A more mature perspective is not which method is best; but when is a method the best. The method and the methodology utilised must first be chosen with great and deliberate care to minimise wasted work and fruitlessness. This is

reinforced by (Bennett et al 2002, p 567), who wrote that 'many factors affect the appropriateness of a methodology, including type of project (large, small, routine or mission critical), application domain (real time, safety critical, user centred, highly interactive) and nature of the Information Systems development organization'.

One of the major influences on the quality of the systems developed is the software development method adopted. If the approach used is not appropriate for a particular type of application then it may limit the quality of the system being produced, (Bennett et al, 2002, p57).

Software Methodologies can be grouped into soft and hard methodologies and further into sub-grouped into structured methodology and object oriented methodology. 'The object-oriented approach provides a mechanism for mapping from real-world problems to abstractions from which software can be developed effectively. (Bennett et al, 2002, p57) continued to assert that object orientation provides conceptual structures that help to deal with modelling complex information systems.

As information systems requirements are becoming increasingly complex, the use of an object-oriented approach is more necessary. It is a sensible strategy to transform the development of a large, complex system into the development of a set of less complicated sub-systems. Object orientation also aims to provide a mechanism to support the reuse of program code, design and analysis models'.

We will begin with the distinction between soft and hard methodologies. Soft and hard approaches to software development are not to be perceived as competing methods. The soft approach complements the hard approach. It allows a broader view of systems development. As (Flynn,1998 p 325) advocates, it is concerned with what may be broadly termed the environmental effects of information system. That is, it is concerned with the relationship between such systems and social, economic, legal and psychological aspects of the environment.

(Bennett et al, 2002, p 568) asserts that 'soft and hard methodologies cover different parts of the life cycle. In this view, a soft methodology is more useful in the earlier stages of the life cycle, particularly when there is uncertainty about the goals or strategy of the organization as a whole. A hard approach will be more appropriate

once any initial uncertainties and ambiguities have been resolved, since the emphasis then shifts to a specific project with relatively clear goals and boundaries.'

There are two very critical problems currently remaining in software development that the hard, scientific approach has not solved. One is that the current methods used in developing the systems persistently solve the wrong problem. The other is that the current development methods used neglect the wider organizational context (Flynn, 1998 p 333).

The soft approach deals with these problems and addresses them. (Dobson & Strens, 1994) asserted that, a hard approach assumes that the problem to be solved is logically based and has a solution in a computer system, thus limiting the range of problems that can be addressed to those that possess a mathematical or logical solution. Another assumption is that the computer-based solution may be placed in the organization without taking account of the social and psychological context within which the system will interact. As (Bennett, 2002, p556), said 'in practice, methodologies vary widely in philosophy, in completeness of definition or documentation, in coverage of the life cycle, and even in the type of applications to which they are best suited'.

The software development method used is vital to the success of the whole venture and its use has to be judged according to the problem to be solved.
Methodology can be tested only in conjunction with a problem to which it is applied as asserted by (Checkland, 1981, p242). We must take, not methodology, but methodology plus problem and ask, not about the methodology, but about the problem. Was the problem solved?

Hard Methods

These hard methods or approaches can be further subdivided into structured methods and object oriented methods. Structured methods include the following.

Structured Systems Analysis

(Bell et al, 1992) said that 'methodologies based on the structured systems analysis method tend to focus on information movement and analyses. This analysis is broken down in terms of flows, processes, files, sinks and sources. Possibly one of the best examples we have of this approach is the seven-step model designed by (DeMarco. The seven step model is:

1. building a current physical model
2. building a current logical model from the physical model
3. building a logical model of the system to be built consisting of data flow diagrams, a data dictionary, and process specifications
4. creating a family of new physical models
5. producing cost and schedule estimates for each model
6. selecting one model
7. packaging the specification

Linear Sequential

This is sometimes called the 'classic life cycle' or the 'waterfall model'. Pressman, 1997 p 33 said the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and maintenance'

The linear sequential model has the following:

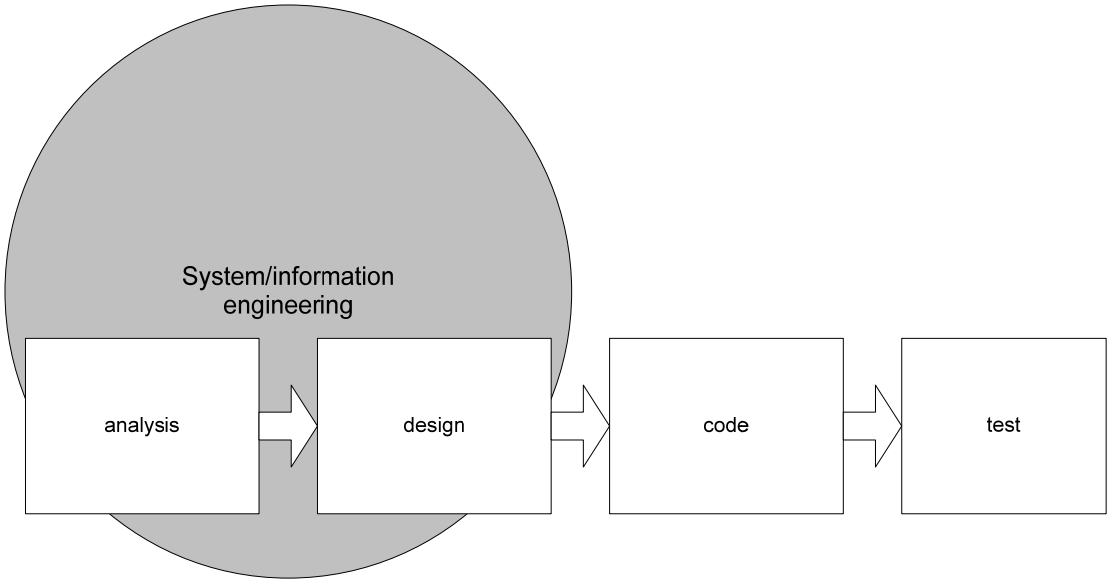


Figure 1: Basic Software Development Life Cycle

Software Requirements Analysis

This involves the gathering of the requirements of the user. It is initiated when the developer tries to acquire an understanding of the problem domain. Several techniques are used in this requirements elicitation stage and it calls for effective communication between developer and user.

Design

The design process translates requirements into a representation of the software that can be assessed for quality before code generation begins. Like requirements, the design is documented and becomes part of the software configuration, (Pressman, 1997 p 34).

Code generation

The design is then transformed from a higher level of abstraction to a more low level machine readable format

Testing

This exposes the code's inherent errors. Testing is usually carried out in a piecemeal fashion going from functional area to module and area prior to testing it as an integrated and whole unit.

Maintenance

This involves adding value to the finished product over the course of its life by patching and amending to increase its overall smooth working and efficiency.

Criticisms of the linear sequential model

The linear sequential model is one of the best known and most popular model for software development. There are however some flaws that have been sighted by users of this classical model. In a real life environment, it is difficult for the software project activities to proceed in a strictly linear manner. The sequential model is not flexible enough to accommodate any contingencies or eventualities. The nature of software development lends itself more to unfolding in iterative fashion. There is some measure of iteration involved, though in an indirect way. Requirements are not usually all garnered at the beginning of the software development. This is because the requirements are not always very clear at the beginning of the software

development project and users are prone to changing their minds or not knowing what they want. One of the drawbacks of the linear sequential model is that it needs all the requirements explicitly at the beginning of the project before it proceeds and there is no mechanism to take more requirements on board as and when they arise. Owing to the linear nature of the model, it takes a very long time for a product to be seen. If there is a major flaw in the product, this could prove fatal to the project as everything would have to start from the beginning again. Not a very efficient way of doing things. In addition, the development team is usually assigned interdependent tasks and if one team member has to wait on another; the wait time may exceed time spent being productive. Despite all the discussed flaws, the linear sequential model holds a significant position in the annals of software development and is still widely used in software engineering.

Prototyping Model

Prototyping allows the user to see a mock up version or a replica of the software system before it is finished. It can be done using tools like Netbeans and Visual Basic. Just the externals of the system are simulated, but the core is missing. This enables the user to have some idea of what the system will look like before it is finished. The prototype can be thought of as the initial system that can serve as the inspiration and motivator for the project before it is thrown away.

Rapid Application Development (RAD) Model

Rapid Application Development (RAD) is a high speed adaptation of the linear sequential model in which rapid development is achieved by using a component-based construction approach. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a 'fully functional system' within a very short time periods (for example, 60 to 90 days) (Pressman, 1997 p 37).

Criticisms of Rapid Application Development (RAD)

If the appropriate number of persons are not present on a RAD team, the likelihood of failure is great. It also needs clients and team members who are committed to very fast paced development and who understand the common vision. Not every software development project is suitable for RAD. (Pressman 1997, p38) said, if a system is not properly modularized, building the components necessary for RAD will be problematic.

Incremental Model

This model is a combination of linear sequential model and prototyping. It could be thought of as systems development in a modularised way. The entire product is sub-divided into smaller portions of the products or increments of the whole product. Linear sequential model is used to deliver and develop each increment. It is useful when the staff complement is not large and therefore delivering the sub-products on time is more manageable. Each increment is prototyped and shown to the client until the whole product is finished.

The Spiral model

This model utilises the strengths of both the linear sequential and the prototyping model. It provides the potential for rapid development of incremental versions of the software. (Pressman, 1997 p42). This evolutionary process model was originated by Barry Boehm.

The Spiral Model is split into several framework activities called task regions. There are on average 3 to 6 task regions. These include customer communication, planning, risk analysis, engineering, construction and release and customer evaluation.

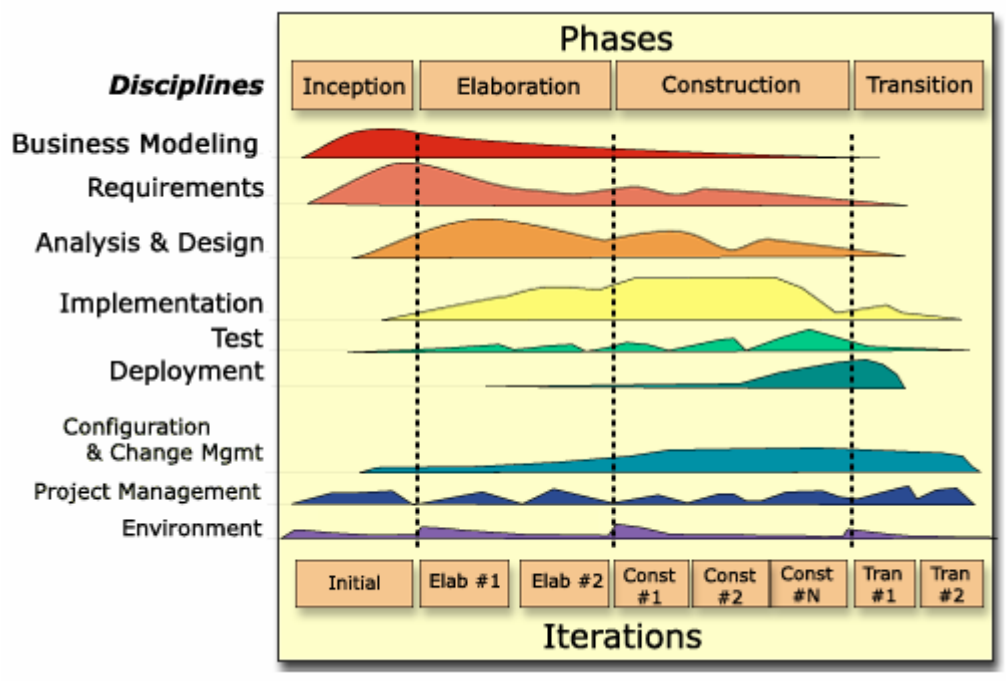
Object-Oriented Analysis and Design Development Processes based upon the UML

A software development process describes how to develop, operate and support one or more software systems. There are several software development processes in the industry, prior and current ones. These include the object-oriented software process (OOSP), the Unified Process, the Microsoft Solutions Framework (MSF), the OPEN Process, eXtreme Programming and Catalysis.

Rational Unified Process (RUP)

Rational Unified Process (RUP) is a system development process produced by IBM Rational. This process is serial in the large and iterative in the small, delivering incremental releases over time, while following proven best practices. RUP is comprised of four phases: Inception, Elaboration, Construction and Transition.

These phases execute sequentially. The project jobs are grouped into logical activities called disciplines. The disciplines are performed iteratively throughout the four phases



Rational Unified Process diagram

"RUP was never intended to be a silver bullet that organizations should apply as is. IBM Rational clearly advocates that organizations customize it to create a process that is specific to meet their particular needs", (Ambler, S, 2002)

Unified Software Development Process (USDP)

USDP is an industry standard generic software development process. It is the iterative and incremental software engineering process for the UML. It has to be customised for use in each development project. USDP is the use-case and risk-driven, iterative and incremental and architecture software engineering process for the UML. USDP has four (4) phases. These are inception, elaboration, construction and transition. Each phase may have one or more iterations. Each iteration has five iteration workflows. They are requirements, analysis, design, implementation, test. Iterations are indispensable to the USDP. Each iteration includes planning, analysis and design, integration and test, internal or external release. These iterations are

organised into phases and contain workflows. A sequence of iterations result in a final product release. According to Ambler, 2001, p 443, the Unified Process has several strengths. First it is based on sound software engineering principles such as taking an iterative, requirements-driven, and architecture based approach to development. Second, it provides several mechanisms, such as a working prototype at the end of each iteration and the go/no-go decision point at the end of each phase, which adds management visibility into the development process'.

eXtreme Programming

Beck, 2000, p xv advocates that 'XP is a lightweight methodology for small-to-medium-sized teams developing software in face of vague or rapidly changing requirements'.

As outlined in Beck, 2002, p xvii, 'XP is distinguished from other methodologies by

- its early, concrete, and continuing feedback
- its incremental planning approach
- its ability to flexibly schedule the implementation of functionality
- its reliance on automated tests written by programmers to monitor the progress of development
- its reliance on oral communication, tests, and source code to communicate system structure and intent'.

The Microsoft Solutions Framework (MSF)

As defined by (Ambler, 2001, p 448), 'MSF is a collection of processes, principles and practices that helps organizations be more effective in their creation and use of technology to solve their business problems. MSF does this by providing rigorous guidance that is flexible enough to be adapted to meet the needs of the project and the organization. Originally based on best practices within Microsoft product development and IT organizations, MSF was created in 1994 and developed into standardized training courses to promote consistency and effectiveness within the Microsoft Consulting Services'.

The OPEN Process

The OPEN process as described in (Ambler, 2001, p 449), is 'a comprehensive software process. The OPEN Process is aimed at organizations using object and component technology…Also similar to the Unified Process, OPEN was initially created by the merger of earlier methods: MOSES, SOMA, Firesmith, Synthesis, BON, and OOram. The OPEN Process supports the UML notation, and any other OO notation to document the work products the OPEN Process produces'.

Criticism of the fusion or unification approach to methods

(Bouzeghoub et al, 1997, p 101-102), highlights the fact that 'a study of the object-oriented world, covering the market and the literature, reveals the existence of a large number of methodologies…..none of the existing methods covers the whole of the project life cycle. Consequently, users are always forced to complete the method in some ad hoc fashion, or to achieve their aims by combining several methods…there are things to be said…..against this tendency to fuse or unify methods…..the fusion or unification approach tends to encourage belief in the existence of a single methodology that will deal with every type of problem when the reality is more complex, and it is more likely that different types of method are better adapted to different types of problems'.

The fusion approach to methods is examined more comprehensively in Chapter 4. Having gained a better understanding of method building in general, the next section highlights additional 'soft' approaches to software development that currently exist.

Alternative Soft Approaches to Systems Analysis and Design methodologies

General systems theory

(Bell et al, 1992, p 186) said that ' this [GST] is a troublesome theoretical perspective to put into practical application, indeed it is not really intended for practical systems analysis: General Systems Theory is too generalised for information systems definition'

Client Led Design

This approach promotes the user directed initiative where the users are in charge of the development process

Critical Systems Thinking (CST)

(Bergvall-Kareborn, B, 2002, p 473-474) wrote that 'Critical Systems Thinking as defined mainly by scholars at Hull University..evolved out of a critique of traditional management science for being positivistic and out of a critique of SSM for its interpretivistic stance….In order to manage situations characterized by conflicting interest groups and get a more democratic approach, CST suggests a philosophy which rests on three "commitments". These are commitments to critique, to emancipation and to pluralism'.

Multi-Modal Methodology (MMM)

According to (Bergvall-Kareborn, 2002), 'MMM is rather new to the field of systems thinking…MMM criticizes the narrowly focused, technological determinism used by the hard systems approaches in dealing with human problems, as well as the soft approaches where the assumption about reality is based on chaos and complexity. Instead MMM…suggests that there is order within complexity'.

Multiview

Uses soft systems as its prior stage before advocating the other stages of software development. It uses a contingency approach where skills of developers and the problem situation are examined before deciding how to proceed with development.

Participative Systems Design

This approach was pioneered by Enid Mumford who argued for a socio-technical approach. She proposed the ETHICS method where the users were major drivers of the development process. QuickETHICS is also popularly used as front end to the process

Criticisms of Soft Approaches

1. They only cover a small part of the life cycle
2. They require more resources

3. Participation may not improve system quality
4. They have not been tested sufficiently in practice
5. Concepts like usability are hard to operationalize
6. Quality problem is not really addressed
7. Impact of new technology on old requirements

Gap in the literature

Avison et al, 1990, p 268 disclose that 'the Multiview methodology is in a continuing state of development….all information systems development methodologies have limitations…We expect the methodology to improve in the future. This research builds on the work done in Multiview methodologies and as the authors have disclosed, there is room for improvement.

2.11 Conclusion

SSM was chosen as the soft method of choice for several reasons. One was for its proven efficacy. As the case studies in this chapter showed, SSM has been used effectively in many cases in conjunction with other methods to develop successful systems. Another reason was that the attributes of SSM such as its ease of use and flexibility made it easy to combine with the UML based method. The outcome of any use of the SSM methodology will be a new problem situation. This is so as the methodology itself is an ongoing learning system whose task is really never done because learning can never truly be complete (Checkland, 1981, p 237). The effectiveness of the SSM becomes greater when combined with another methodology. Methodological Pluralism or multi-methodology, whatever name it goes by is here to stay. It is finding its niche. As it evolves into a more stable discipline with firm theoretical and philosophical underpinnings, it is still being used to ensure radical benefits and advantages in the software development arena. This is explored in more detail in Chapter 4. In this research the SSM method is combined with a development process based on the Unified Modelling Language (UML). This produces the resultant MoIST method. The UML consolidates a set of core modelling concepts that are generally accepted and used in conjunction with many current methods. These methods include RUP and USDP. The UML provides users with a ready-to-use, expressive visual modelling language. This can then be used to develop and exchange meaningful models.

Chapter 3- Unified Modelling Language (UML)

3.1 Introduction

Whereas Soft Systems Methodology represents unstructured situations, the Unified Modelling Language represents the 'harder' information systems paradigm. One of SSM's primary goals is to more accurately understand an unstructured environment as it really is. The results of this more clearly understood situation is then fed into the UML based development process. This helps to further define the existing situation and clarify solutions. At this point of the process this is where the UML's applicability is recognised and comes in.

The UML notation is ideal as a follow on from the use of SSM. This is because the characters represented within the rich picture and subsequent conceptual models find a related progression in the UML's use cases with its own characters and then class, sequence and other diagrams. This provides some measure or degree of cohesion and logical flow from architecture by SSM to implementation and realisation through UML. Soft and hard methodologies cover different parts of the life cycle. In this view, a soft methodology is more useful in the earlier stages of the life cycle, particularly when there is uncertainty about the goals or strategy of the organization as a whole. A hard approach will be more appropriate once any initial uncertainties and ambiguities have been resolved; insofar as this is possible, since the emphasis then shifts to a specific project with relatively clear goals and boundaries (Bennett et al, 2002, p568).

3.2 History of the UML

Prior to the UML, there was no clear leading modelling language. Users had to choose from among many similar modelling languages with minor differences in overall expressive power. Most of the modelling languages shared a set of commonly accepted concepts that are expressed slightly differently in various languages. Users longed for the industry to adopt one, or a very few, broadly supported modelling languages suitable for general-purpose usage.

(OMG, 2003)

The UML had its beginnings in the late 1980s (Booch et al, 1998).[3] The UML was originally conceived by Rational Software and three of the most prominent methodologists in the information systems/technology industry: Grady Booch, James Rumbaugh, and Ivar Jacobson. It represents the evolutionary unification of their experience with other industry engineering best practices. Faced with new object-oriented programming languages and increasingly complex applications, methodologists began to experiment with alternative approaches to analysis and design. The number of OO methods increased from fewer than 10 to more than 50 between 1989 and 1994. Many users of these methods had trouble finding a modelling language that met their needs completely, thus fuelling the so-called methods wars. As users learned from experience, new generations of these methods began to appear, a few clearly prominent, most notably Booch, Jacobson's Object Oriented Software Engineering (OOSE), and Rumbaugh's Object Modelling Technique (OMT).

The UML effort started officially in October 1994, when Rumbaugh joined Booch and OMT methods. The version 0.8 draft of the Unified Method (as it was then called) was released in October 1995. Jacobson then joined Rational and the scope of the UML project was expanded to incorporate OOSE. This resulted in the release of the UML version 0.9 documents in June 1996. The general software engineering community was invited to give feedback to the UML effort.

The three of us started the UML effort at Rational and were its original chief methodologists, but the final product was a team effort among many UML partners under the sponsorship of the OMG. All partners came with their own perspectives, areas of concern and areas of interest. This diversity of views strengthened the final result. We expect that OMG's ownership of the UML standard and the public's free access to it will ensure the widespread use and advancement of UML technology over the coming years.

(The three amigos, OMG, March 2003)

In September 1997, the OMG officially adopted its first methodology standard, the Unified Modelling Language. Twenty one companies participated in the momentous effort to create a single, standardized analysis and design notation and metamethodology. In the subsequent years, the UML standard has successfully

[3] Booch, G, Rumbaugh, J, and Jacobson, I. The Unified Modelling Language User Guide. Addison Wesley, Reading, Mass., 1999.

unified the previously highly fragmented object-oriented analysis and design industry. The OMG in an unprecedented move successfully drew the community together to agree a single, worldwide standard. This achieved even higher consensus than OMG's successful CORBA systems integration platform standard (Henderson-Sellers et al, 2000).

Many software organizations saw the UML as strategic to their businesses. A UML consortium was formed. This had several organizations willing to dedicate resources to work towards a strong and complete definition. A semantics task force was formed to integrate the UML with other standardization efforts. A revised version of the UML 1.1 was offered to the OMG for standardization. This version was accepted by the OMG. UML 1.1 was adopted by the OMG. Maintenance of the UML was then taken over by the OMG Revision Task Force (RFT). In June 1999, the RTF released UML 1.3.

The major UML 2.0 revision has improved the UML's semantics of extension by profiles. The Systems Modelling Language (SysML) is a domain-specific modelling language. It is used for systems engineering and is defined as a UML 2.0 profile

UML is a general-purpose modelling language that has a standardized graphical notation. This notation is utilised in the generation of an abstract model of a system. This model is a UML model. One of the characteristics of the UML is its extendability. For this profiles and stereotype are used to achieve customization.

UML frees software developers to focus more on design and architecture. UML does this by creating an industry standard graphic notation. The notation represents common concepts such as classes, components, aggregation, behaviours and generalization.

The OMG's official definition of the UML is the UML meta-model. This is a Meta-Object Facility meta-model (MOF). The UML meta-model and UML models including all MOF-based specification may be serialized.

The UML has become part of the mainstream of software development, enabling various stakeholders, to gain control of their systems architecture, and to manage complexity.

The UML represents the culmination of best practices in practical object-oriented modelling. The UML is the product of several years of hard work that were focussed on bringing about a unification of the methods most used around the world, the adoption of good ideas from many quarters of the industry, and above all, a concentrated effort to make things simple (OMG, 2003).

3.4 Overview of the UML

The Unified Modelling Language (UML) is a graphical notational language. The UML is used to visualize, specify, construct and document the artefacts of a software intensive system. It is a software standard that is owned by the Object Management Group (OMG). UML is a public domain modelling language that is available to anyone and everyone who wishes to follow a disciplined and standard approach to modelling systems and applications (Henderson-Sellers et al, 2000).

The UML has become part of the mainstream of software development, enabling various stakeholders, to gain control of their systems architecture, and to manage complexity.

The UML represents the culmination of best practices in practical object-oriented modelling. The UML is the product of several years of hard work that were focussed on bringing about a unification of the methods most used around the world, the adoption of good ideas from many quarters of the industry, and above all, a concentrated effort. The UML functions as the means for expressing and communicating knowledge. The UML brings together the industry's best practices regarding how we understand the world around us and how we represent and communicate that understanding. It has four distinguishing characteristics in comparison to other modelling languages: It is general-purpose, broadly applicable, tool-supported, and industry standardized.

Among its other benefits, is the market share held by industry and tool vendors supporting it, widespread use of the methods founded by its creators, and its adoption by the OMG will make the UML a pivotal force in today's businesses. (OMG, 2003).

Diagrams of the UML

The UML provides mechanisms for organizing and classifying knowledge regarding a given context or situation in which the problem resides and in which the solution must be implemented. Such knowledge is captured in a model consisting of various modelling elements, and it is represented through distinct but interconnected sets of diagrams. The model itself captures the knowledge, and the diagrams represent the knowledge in a communicable form. A model is an abstract representation of a specification, a design or a system, from a particular point of view. It is often represented visually by one or more diagrams. It aims to express the essentials of some aspect of the process without giving unnecessary details. Its purpose is to enable people involved in the development to think about and discuss problems without getting sidetracked. Modelling is simply a form of abstraction that facilitates both problem understanding and problem resolution (Selic, B 1999)[4]. A model captures only the significant features of a system and hides or ignores lower-level detail. The simplified yet relatively accurate view of reality presented by a model is much easier to comprehend than the actual system. The choice of what to model has an enormous effect on the understanding of the problem and the shape of the solution. When deciding how to model something, determining the correct abstraction and detail is critical to providing something that will be of benefit to the users of the model.

The UML 2 consists of 13 distinct, but interconnected diagrams through which to present a given body of knowledge. These include use cases, class, sequence, and activity diagrams. Use cases and activity diagrams will be explored in more detail here. This is because they have greater relevance to this particular research.

Activity Diagrams

The activity diagrams are special cases of state diagrams that capture activities or actions of elements. They describe knowledge regarding the behavioural characteristics of the involved elements and the dynamic interactions or collaborations among them. The activity diagrams help us to understand how different entities behave and interact in order to realize their objectives.

[4] Turning Clockwise: Using UML in the Real-Time Domain – Selic, B

UML Activity Diagrams are typically used for modelling the logic captured by a single use case usage scenario. Although UML activity diagrams could potentially model the internal logic of a complex operation it would be far better to simply rewrite the operation so that it is simple enough that you don't require an activity diagram. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structured development.

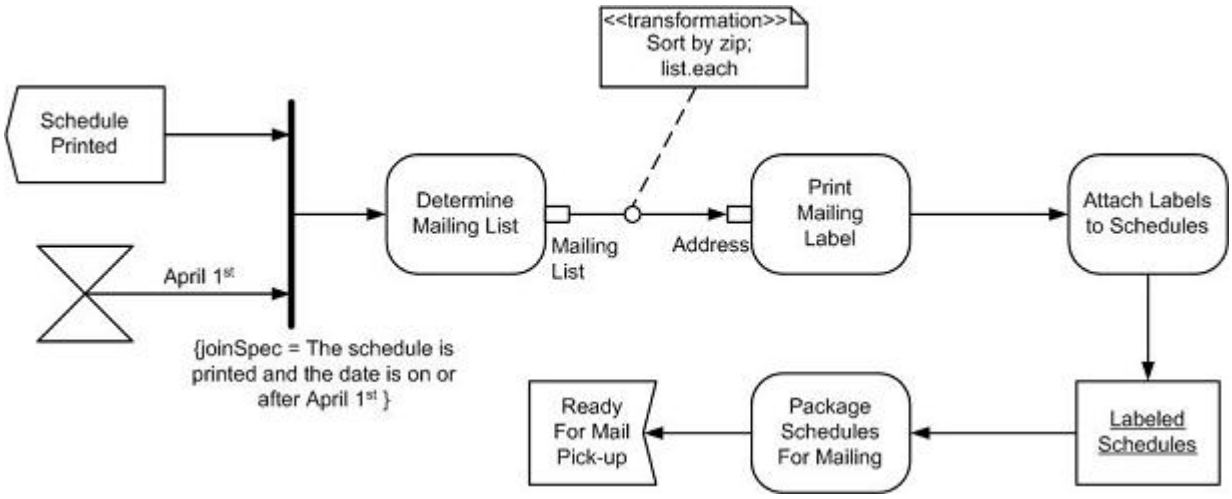


Fig 2:1 Example of an Activity Diagram for a use case to distribute schedules

Each of the diagrams discussed above captures a different set of concerns and aspects regarding the subject, and each modelling element represents some concept, construct, or element of knowledge regarding the subject. These diagrams, with the modelling elements they use, describe the content of the communication among the individuals involved in the problem-solving process. Together all the diagrams holistically form an integrated window to the body of knowledge that is applied and gained through the process. It is the sharing and reapplying of this knowledge (and artefacts representing its fragments) that enables an organization to capitalize on the benefits of applying the UML. Together, these diagrams establish a coherent body of knowledge regarding the business, the problem/solution, and the problem-solving process by addressing and reconciling the concerns of the various stakeholders. This coherence will be shown later in the research and is essentially that there is some element or measure of relatedness among syntactic components such as use cases and activity diagrams.

Basic Activity Diagram notation

- **Initial node**. The filled in circle is the starting point of the diagram. An initial node isn't required although it does make it significantly easier to read the diagram.
- **Activity final node**. The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes.
- **Activity**. The rounded rectangles represent activities that occur. An activity may be physical or electronic.
- **Flow/edge**. The arrows on the diagram.
- **Fork**. A black bar with one flow going into it and several leaving it. This denotes the beginning of parallel activity.
- **Join**. A black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue. This denotes the end of parallel processing.
- **Condition**. Text on a flow, defining a guard which must evaluate to true in order to traverse the node.
- **Decision**. A diamond with one flow entering and several leaving. The flows leaving include conditions.
- **Merge**. A diamond with several flows entering and one leaving. The implication is that one or more incoming flows must reach this point until processing continues, based on any guards on the outgoing flow.
- **Partition**. These are also called swimlanes, indicating who/what is performing the activities.
- **Sub-activity indicator**. The rake in the bottom corner of an activity indicates that the activity is described by a more finely detailed activity diagram.
- **Flow final**. The circle with the X through it. This indicates that the process stops at this point.
- **Note**. A standard UML note that indicates extra information needed. Stakeholders usually find them easier to understand.

General Guidelines for drawing activity diagrams

1. Place the start point in the top left-hand corner. A start point is modelled with a filled in circle. Every UML Activity Diagram should have a starting point.
2. Always include an ending point. An ending point is modelled with a filled in circle with a border around it.

2. Activities

An activity, also known as an activity state, on a UML Activity diagram typically represents the invocation of an operation, a step in a business process, or an entire business process.

1. Question “Black Hole” Activities. A black hole activity is one that has transitions into it but none out, typically indicating that you have either missed one or more transitions.
2. Question “Miracle” Activities. A miracle activity is one that has transitions out of it but none into it, something that should be true only of start points.

3. Decision Points

A decision point is modelled as a diamond on a UML Activity diagram. Decision Points Should Reflect the Previous Activity. The guards on leaving the decision point also help to describe the decision point.

4. Guards

A guard is a condition that must be true in order to traverse a transition.

1. Each Transition Leaving a Decision Point Must Have a Guard
2. Guards Should Not Overlap.
3. Guards on Decision Points Must Form a Complete Set.
4. Exit Transition Guards and Activity Invariants Must Form a Complete Set. An activity invariant is a condition that is always true when your system is processing an activity.
5. Apply a [Otherwise] Guard for “Fall Through” Logic.
6. Guards Are Optional. It is very common for a transition to not include a guard, even when an activity includes several exit transitions.

5. Parallel Activities

It is possible to show that activities can occur in parallel using two parallel bars. The first bar is called a fork. It has one transition entering it and two or more transitions leaving it. The second bar is a join, with two or more transitions entering it and only one leaving it.

1. A Fork Should Have a Corresponding Join. In general, for every start (fork) there is an end (join).
2. Forks Have One Entry Transition.
3. Joins Have One Exit Transition

6. Swimlane Guidelines

A swimlane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread.

The activity diagram shows the steps of a computation. Each step is a *state* of doing something. For that reason, the execution steps are called *activity states*. The flow of control from one activity state to the next is called a *transition* (Maciaszek, 2001). Activity diagrams can have other uses in system development apart from modelling use cases (Fowler and Scott, 2000). They can also be used to understand a business process at a high level of abstraction.

Use-cases

The use-case view describes knowledge regarding the needs and requirements of the various stakeholders. This view is depicted by use-case diagrams. "A use-case diagram captures relationships among various entities and their roles, responsibilities, and objectives within the environment. This view functions as the primary motivating force for the whole problem solving process and provides validation criteria for the resulting solution." A use case is a sequence of events that achieves a measurable result for an actor or it is an example of how someone or something uses the system (Jacobson, 1995).

Use cases document the behaviour of the system from the users' points of view. By 'user' in this case we mean anything external to the system being developed which

interacts with the system. A user might be a person, another information system or a hardware device. Use case modelling helps with three of the most difficult aspects of system development:

- Capturing requirements
- Planning iterations of development
- Validating systems

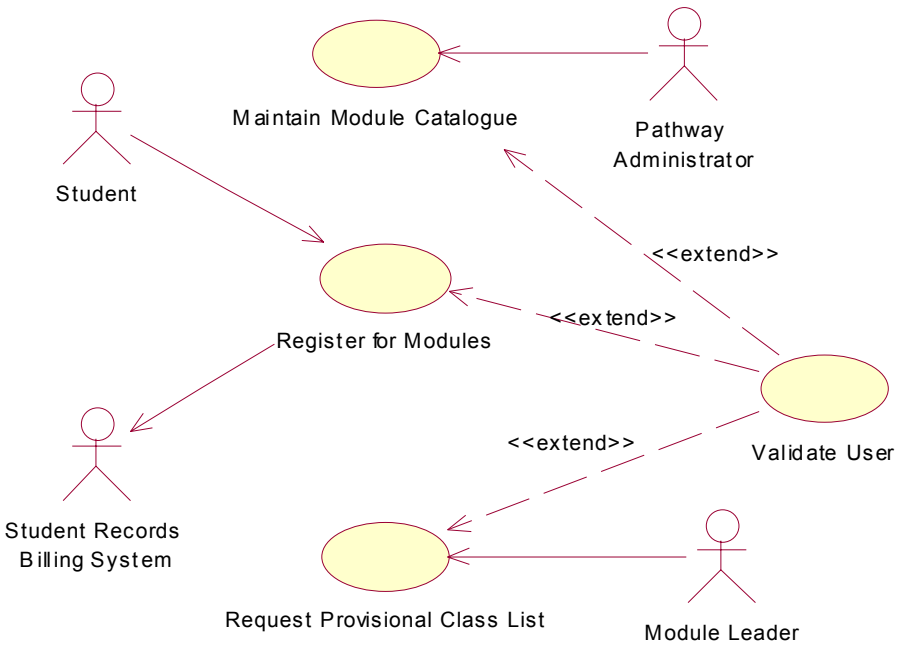
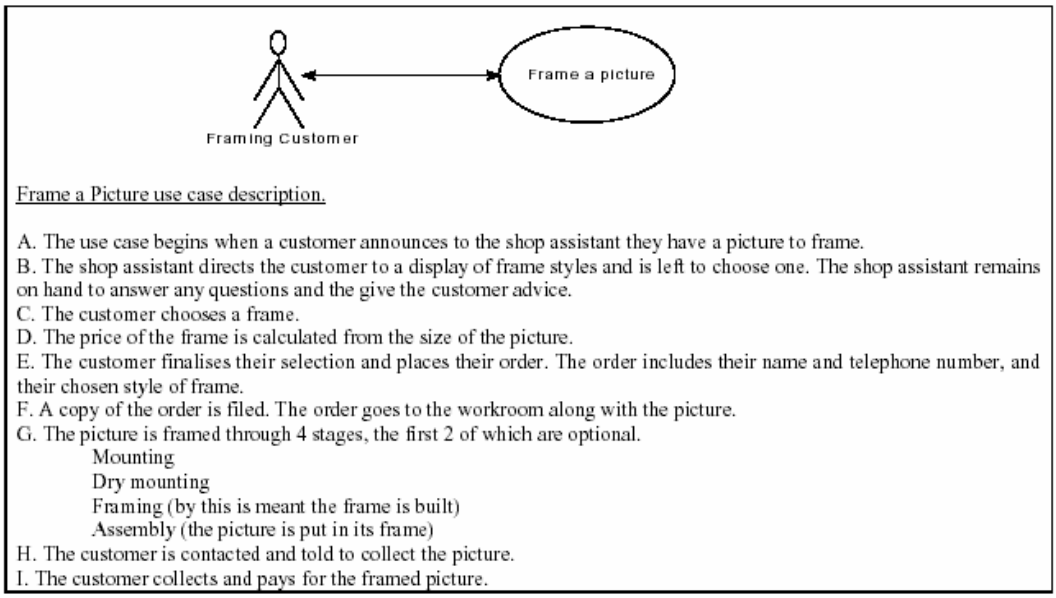


Figure 3.1: Use cases for Module registration

History of use cases

Use cases were first introduced by Ivar Jacobson in the early 1990s as a development from the earlier idea of scenarios. These scenarios have evolved into what are now known as use cases. A use case diagram is comparatively easy to understand intuitively, even without knowing the notation. This is an important strength, since the use case model can sensibly be discussed with a client who need not be familiar with UML.



Frame a Picture use case description.

A. The use case begins when a customer announces to the shop assistant they have a picture to frame.
B. The shop assistant directs the customer to a display of frame styles and is left to choose one. The shop assistant remains on hand to answer any questions and the give the customer advice.
C. The customer chooses a frame.
D. The price of the frame is calculated from the size of the picture.
E. The customer finalises their selection and places their order. The order includes their name and telephone number, and their chosen style of frame.
F. A copy of the order is filed. The order goes to the workroom along with the picture.
G. The picture is framed through 4 stages, the first 2 of which are optional.
Mounting
Dry mounting
Framing (by this is meant the frame is built)
Assembly (the picture is put in its frame)
H. The customer is contacted and told to collect the picture.
I. The customer collects and pays for the framed picture.

Fig 3.2: Use case model to frame a picture, Chesney and Fletcher, 2000

A use case model consists of three (3) main components. These are actors, use cases and relationships. A use case model is created at the beginning of systems development to capture system requirements.

An actor initiates a use case, and an actor (possibly the initiator, but not necessarily) receives something of value from the use case. The graphic representation is straightforward. An ellipse represents a use case, a stick figure represents an actor. The initiating actor is on the left of the use case, and the receiving actor is on the right. The actor's name appears just below the actor. The name of the use case appears either inside the ellipse or just below it. An association line connects an actor to the use case, and represents communication between the actor and the use case. The association line is solid, like the line that connects associated classes. The actors, use cases, and interconnecting lines make up a *use case model*,(Schmuller, 1999).

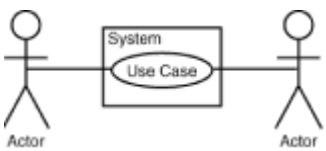


Figure 3.3: Use case model, Schuller, 2004

Actors:

An actor, usually shown as a stick person, represents a kind of user of the system (by user we mean anything external to the system that interacts with it.

Use Cases:

An individual use case, shown as a named oval, represents a kind of task, which has to be done with support from the system under development. The UML standard calls this a 'coherent unit of functionality'. Of course the use case diagram shows only a small part of the information we need. Each use case is described in detail, usually in text. The use case diagram can be seen as a concise summary of the information contained in all the descriptions of the use cases

Relationships:

This describes or depicts the way that use cases relate to each other. They provide a link between actors and use cases. Actors use use cases and use cases can use other use cases. Relationships are depicted as lines sometimes with arrows. A relationship speaks of a two (2) way communication and the arrow direction is an indication of the interaction initiator.

A use case can use another use case. If you have a piece of well-defined functionality, it makes sense to re-use this wherever possible. Also, sometimes a use case gets too big to manage sensibly and it makes sense to break this down into smaller use cases. There are two ways use cases can relate. The first is where a use case "includes" another use case. In this case the second use case is always invoked as part of the execution of the first. This is drawn with an arrow pointing to the use case that is included, with the label <<include>> tagged to the line. So the following diagram shows that we always include provision of the module timetable as part of the task of enrolling a student on a module (Wade et al, 2002).

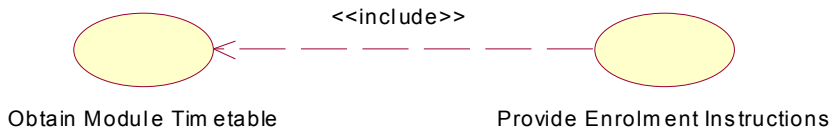


Figure 3.4: Illustration of the 'Include' Label

Sometimes a use case is only called occasionally from another use case. This will often be to support an alternative path or an exception. We draw this with an arrow pointing the other way where the arrow points to the calling use case. Thus the following diagram says that it is sometimes (but not always) necessary to extend the process of enrolling a student by chasing up their previous qualifications.

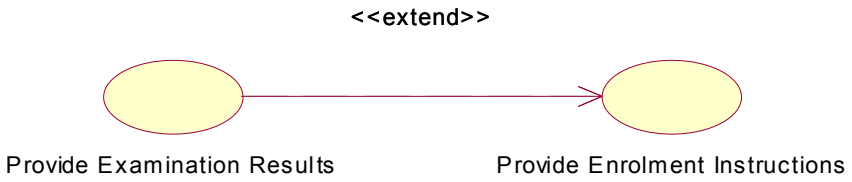


Fig 3.5: Illustration of the 'extend' Label

Justification for using UML in the research

The whole aim of the research is to make software development easier and more in line with what the client needs. SSM is already user focussed and as expounded in Chapter two it offers a greater chance of satisfying client requirements and therefore a more substantial chance of a successful system. The other aspect to making software development more accurate and successful involves the use of the UML. The big advantage that use cases have over other requirement models is that it is an excellent tool for communication between developers and users. This is as it is written in the user's language and requires little or no knowledge of a modelling notation to understand (Chesney and Fletcher, 2000). The advantage of Use Cases in the research is further highlighted in Chapter 6 – the intervention chapter. There it is seen that the communication between SSM methodologist/developer and client was achieved satisfactorily. In the event that this had not been the case, a second option would have been to utilise the use cases developed in applying the MoIST method to the situation to further enhance the client understanding of the research findings. The client in this research – the academic support tutor – represented management. Additional research has shown that management who are also end users of the system can also get involved in systems development. This is as the use case model would be created by the problem domain experts – the management. They would be guided by the software developers. This is usually more accurate and could be used by developers as a starting point for analysis. Other end users would be involved in validating the models and this would mean that

users would be less likely to resist the new system as they would have been involved in its development (Chesney et al, 2000, Sauer, C, 1994).

Limitations of with use cases

1. Focussing on use cases may encourage developers to lose sight of the architecture of the system and of the static object structure, in the rush somehow to deliver the use cases which are required in the current iteration.

2. There is a danger of mistaking design for requirements. More generally, requirements by use cases may encourage developers to think too operationally: users are likely to describe the use case as a very concrete sequence of interactions with the system which is one way, not the only way of achieving their goal. It is important that developers distinguish between requirements and candidate design.

3. There is danger of missing requirements if too much reliance is `put on the suggested process of finding the actors and then finding the use cases that each actor needs. Not all requirements emerge this way. This danger can be lessened by doing use case analysis and conceptual class modelling in parallel.

3.5 UML Benefits and goals

- Provides users with a ready-to-use, expressive visual modelling language to develop and exchange meaningful models. The UML consolidates a set of core modelling concepts that are generally accepted across many current methods and modelling tools.

- Furnishes extensibility and specialization mechanisms to extend the core concepts. Though the core concepts cannot be changed by the users, UML allows the users some leeway. Users are allowed to build models using core concepts without using extension mechanisms for most normal applications.

They can also add new concepts and notations for issues not covered by the core.

- UML is an expressive language. It is therefore possible to express important aspects of the design and meaningfully to reflect changes in the design, which are made during the development as changes in the models.

- UML is supported by suitable tools, so that the developer's efforts can be spent on work that requires their skills, not on routine work.

- When new people join the project, it is an advantage if they already know the modelling language instead of having to learn it then.

- To do component-based design, one has to be able to read the descriptions of components. The more easily and quickly this can be done, the cheaper it is to consider a component. The more widely used the modelling language, the greater the chance that it is the same one, the component writer will choose to use.

Limitations of the UML

- The UML has become a necessary part of most software activities. Nevertheless it is not sufficient. Insufficiencies of the UML arise from the fact that it is a pure modelling language, nothing more. It contains no elements of process that would guide software development from 'start' to 'finish'. Being only a modelling language, it also does not take responsibility for other issues such as data issues and project management issues. What is needed is a customisable process where the process could be tailored to fit the precise needs of each organization.

- There is an unclear semantic description of the UML syntax.

- There is a risk of circular definition with the UML. This means that the target language is not rich enough to define itself and can ultimately lead to repetition.

3.5 Conclusion

The Unified Modelling Language (UML) paradigm is here to stay. Over several years it has grown from being a feature of a relatively obscure graphical notational language to become a general, universally accepted technique that offers a uniform approach to software development from analysis through to coding (Bustard et al, 1994).

Chapter 4- Successful Integration of Systems Thinking into IS development

It is currently beyond the scope of use cases to help with the important analysis of how and where information technology could improve on an existing process (Butler, 1998)

4.1 Introduction

The UML has been introduced as a diagram-based language for describing designs. “Diagrams are how we naturally think about systems” (Buzan and Buzan, 2000). “It is inconceivable that a single diagram could capture everything about a design. That is indeed not desirable, as we will be interested in different aspects of the design at varying times” (Steven & Pooley, 2000). Integrating systems thinking into information systems development helps to give a more comprehensive and balanced view or picture of the problem and its solution.

Now, more than ever, planning and managing in the real world is beset by change and uncertainty. Knowledge is incomplete, values are in dispute, decisions of others are often unpredictable. Sheathed in opaque technicalities, inflexible and over-ambitious, the highly mathematical methods of analysing problem situations are no longer considered acceptable. In their place a coherent alternative paradigm has emerged. This is a range of methodologies which aim not to produce 'optimal' solutions but to facilitate an enriched decision-making process. ‘Low-tech' transparent and participatory, these methods assist in the formulation and reformulation of problem solving in an uncertain world (Rosenhead et al, 2001).

One of the main challenges of undertaking IS design is the need to find some means of moving from methods of inquiry suited to sense making in social situations, to methods suited to organizing knowledge into a suitable format for the construction of a logical specification for any supporting technology (Champion and Stowell, 2002). Many methods and approaches have been innovated over many years. There is however no magic formula for getting the most accurate requirements specification whilst simultaneously considering all the socio- technical factor then churning out the logical blueprint for a system. There is need for balance. Soft Systems Methodology (SSM) is neither by design nor intent an official software development technique. Its use over the years has proven quite effective in providing structure to unstructured situations.

There have been many writings about work done in SSM by several authors. These authors have advocated the use of SSM in the software development process (Checkland, 1981, Checkland and Scholes, 1990; Wilson, 1990; Checkland and Holwell,1998). Others have taken this work a step further by writing about how to link SSM with information systems development (Miles, 1988; Stowell, 1985; Miles, 1992; Checkland, 1988; Mingers, 1988; Checkland and Scholes, 1990; Mansell, 1990; Prior, 1992; Stowell et al, 1990; Doyle and Wood, 1991; Jayaratna, 1994; Sawyer, 1991; Miles, 1992; Savage and Mingers, 1996; Stowell, 1995). Much energy has been exerted in the past to link SSM with SSADM and its resultant DFD diagrams in the structured design process (CCTA, 1993). Not too much extensive work has been done with linking SSM to the Unified Modelling language (UML) paradigm. However, exceptions to that are (Savage and Mingers, 1996, Bustard and Lundy, 1996).

This research has been influenced by the foundational work that they have done and has resulted in a novel method called MoIST. This thesis linkage is between Soft System Methodology (SSM) and Unified Modelling Language (UML) via the MoIST Method. This discussion is approached cautiously because the literature on the subject is wide ranging and there is no single coherent view of what should constitute best practice. This section provides a historical timeline of how the linkage attempts have fared over the years. It also documents the innovators behind them. It looks at some uses of SSM in information systems development, SSM and Structured Design methods, SSM and Object oriented (OO) methods. It also examines several of the limitations of SSM in information systems development.

4.2 Problems with existing IS methods

Traditional approaches to system design have been defined as ‘product oriented’. This meant that the software product was perceived as being ‘fixed’ and well understood. This led to the product requirements being stated way in advance of design and implementation. Information strategy formation is no longer seen as the sole remit of senior management. There is now the clamour for employee involvement as even with the right conditions, the learning process will continue after the technical implementation of an information system, since results concerning use can never be fully predicted during system design (Walsham, 1993).

Usually computerised information systems follow the approach of analysis of information requirements, construction and implementation of a system. This fixed

approach has been to the exclusion of systems thinking. It could be said that the field of information systems has neglected systems thinking as an underpinning to both its theoretical and practical concerns (Checkland, 1988). This may be because the managers feel that introducing an information system into an organization raises more social issues than technical ones and its employees are the ones who would have to make the effort to adjust to this new modus operandi. The reactions and the complexity of the social issues vary according to 'how' the process of developing and introducing the clients to the information system was done.

Organisations used to be perceived as functional, sterile and non-emotive environments where one just got on with the job at hand. Nowadays, organizations are increasingly questioning their purpose and processes. Boundaries between them have become increasingly fuzzy and vague. The management thinking has now changed to reflect the societal facet of an organisation as a place of conflicts, varying affiliations and emotions (Lai, 2000). This paradigm shift in organisational thinking has heralded the need for a change in the way information systems are designed for organisations. This 'process' oriented approach encourages and champions the inclusion in software development of human activities, communication and learning. The requirements are therefore not predetermined and rigid; but more emergent in nature. This allows the users to input their unique needs to mould the functionality of the software product to fit their particular organizational ethos. It is no longer possible to start with the notion that it is necessary to create or computerize an information system. Information system development has to be seen as a continuous process which is led by the human activity system in the organization which the information system will serve (Lai, 2000; Checkland and Holwell, 1993).

4.3 Justification for combining Systems Thinking with IS

No method or methodology in any discipline is able to offer a complete view of the complexities facing organizations. Each may offer a snapshot that provides insights that are useful for reflection and action. Using two or more methodologies in the same intervention is likely to produce a richer picture for seeing and understanding the complex web of relationships and interconnectivities which is likely to lead to better decision taking by managers and team members in software development projects (Mingers et al, 1997).

This research advocates the practice of combining the SSM and UML methodologies in the life span of an intervention. This practice is also referred to in some circles as methodological pluralism or multi-paradigm intervention and research. It refers to the whole area of utilizing a plurality of methodologies or techniques within the practice of taking action in problematic situations (Mingers, 1997).

The practice of mixing methodologies opens up a large set of options and combinations and permutations. These include methodologies combined in the same interventions and single paradigm combination. The latter combines methods of only the soft or only the hard domain in one intervention. This is in contrast to multi-paradigmic combining. Here mixing is supported across paradigmic domains. Where the methodologies are all from within the same paradigm there is little philosophical difficulty, it is just a question of the most effective way of fitting the methodologies or techniques together. When they are from different paradigms, however, the situation is much more complex (Mingers, 1997).

The practice of mixing methods is at present a fledgling concept gaining more respectability as the years go by. It has been much criticised by the purists among the methodologists who do not believe in what they call hybrid methodology. Despite its naysayers, it has been used successfully in numerous interventions over many years (Lai, 2000, Ormerod, 1995). It is vitally important though for the proper development of the discipline that the theory that envelops the practice be explicit and sound. There must be in depth consideration of the philosophical and theoretic facets of multimethodology, since the practice of combining methods is in regular use (Mingers, 1997). This is because without an explicit theoretical underpinning to their work neither consultants, nor academics and their students, can learn from “pluralistic practice (Jackson, 1997).

In reviewing the literature and while considering this area of the research, I was reminded of how the SSM developed under (Checkland, 1981, 1990). Checkland attempted to solve real world problems with systems engineering methodology. On discovering the futility of doing this and 'stumbling' upon the systemic way, he set about buttressing the novel but growing practice with philosophical and theoretical underpinnings. This led to the seminal work of (Checkland, 1981) and over twenty-five (25) years later, SSM is growing in popularity and its boundaries being expanded by converted practitioners (Mingers, 1997). If one may be allowed to predict happenings within research, I predict a similar future for the practice of combining methods in software development. I expect greater results from using this practice, because unlike how SSM developed, there is a plethora of existing examples that have now become the foundation that theorists and software developers can build on and emulate.

4.4 Previous Related Work done in the area of Systems Thinking and IS Development

The concept and practice of integrating systems thinking into information systems development is not a new frontier. To combine two epistemologically diverse methodologies sounds impossible to some (Butler, 1998). The thesis however shows that this is not the case as for years many practitioners have attempted this and have had varying degrees of success. Notable ones include (Checkland and Griffin, 1970; Miles, 1988, 1992; Avison and Wood-Harper, 1990; Checkland and Scholes, 1990; Galliers, 1992; Checkland and Holwell, 1998; Bustard et al, 1996, Savage and Mingers, 1996, Lai, 2000, Champion and Stowell, 2002, Mingers, 2001).

This section starts with a tabulation of various practitioner efforts. It then goes on to examine methods derived to link SSM to Object orientation and SSM to Structured design methods.

Authors	Comments
Mingers 2001	Efforts to 'front end' SSM onto structured design methods and to embed IS methods within SSM, with SSM guiding the whole project
Doyle and Wood 1991	Show problems that arise from integration because of the different and conflicting epistemologies embodied in SSM and IS methodologies
Avison and Wood-Harper 1990	Represents the longest running attempt to bring together hard and soft approaches to IS development
Watson and Wood-Harper 1995	A more recent perspective on the Multiview approach. This differs from the original multiview perspective done with Avison in 1990. It says that multiview is simply a metaphor for the process of defining an information system.
Ormerod 1995	Uses multimethodology in the development of an information systems strategy for Sainsbury's supermarkets. It used cognitive mapping, SSM and strategic choice in the various phases
Mingers and Brocklesby 1996	Encourages multimethodology as a means of provision of great flexibility in an intervention.

Table 4.1: Tabulation of some of the previous attempts to combine methods

Method	Methodologist	Proximity to MoIST (Close or Not Close)
BASE	Bustard, He and Wilkie	Close to MoIST
BOOST	Dobbin and Bustard	Close to MoIST
CCTA's (SSM + SSADM)	CCTA	Close to MoIST
Client-Led Design	Stowell and West	Not Close to MoIST
Contingency Framework	Davis	Close to MoIST
COT Framework Approach	Checkland	Not Close to MoIST
DSDM	Martin	Not Close to MoIST
ETHICS	Mumford	Not Close to MoIST
Gap Navigation	Stowell and Champion	Not Close to MoIST
Grafting vs. Embedding	Miles	Not Close to MoIST
ISD Framework	Lai	Close to MoIST
Multiview	Avison and Wood-Harper	Close to MoIST
RACE	Bustard and Lundy	Close to MoIST
RAD	Martin	Not Close to moist
Zachman's Framework	Zachman	Not Close to MoIST

Table 4.2: Classification of Multi-methods in terms of their proximity to MoIST

Linking SSM to OO Methods

Most of the work done in extending SSM to information systems design has been with structured design methods. Linking SSM to OO is a relatively new area. Nevertheless, advances are being made as OO increases in popularity in the software development field.

Lai's ISD Framework - 2000

Here a framework which incorporates elements of systems science and object oriented methodology is formulated. This framework links SSM and Martin-Odells's object oriented analysis (OOA). Here the modelling techniques of OOA are embedded within SSM. This is somewhat related to R K Miles' grafting vs embedding approach (Miles, 1988, 1991). Lai advocates the use of systems science and object orientation together in order to increase the effectiveness of organizational requirements analysis for IS development. Lai defines 6 types of gaps that analyse IS failure. They are the cognition, comprehension, expression, delivery, utility and expectation-perception gaps. The products of SSM are used to define a plumb-line for the work done in the modelling phase of OOA. The OOA products are then evaluated via a review process. This is an achievable approach and its success is established by application in a governmental Labour Division in Hong Kong, (Lai, 2000).

MoIST versus ISD Framework

MoIST and ISD Framework have the same goal of linking SSM with a 'hard' method. Each however utilises differing techniques to achieve its goal. The main potential problem with the ISD Framework is that Martin-Odell's OOA has now been subsumed by a UML –based method. OOA is no longer used as much as UML-based techniques. MoIST intends to address this potential problem by using a UML-based method in lieu of OOA. This will serve the purpose of making the linking method more relevant and commercially feasible.

Requirements Acquisition and Controlled Evolution (RACE) – Bustard and Lundy, 1996.

The RACE method describes an integration between formal modelling in LOTOS and the use of less formal descriptions of behaviour in soft systems activity models. RACE was developed at the University of Ulster with David Bustard as a pivotal

project team member. The method integrates process-oriented formal modelling with activity modelling in soft systems analysis. The overall aim is to improve the requirements engineering process. RACE was constructed around SSM. SSM develops activity models. This modelling provides some sort of linkage with process modelling which describes the behaviour of the system. It involves SSM elements such as root definitions, conceptual modelling and defines its own interaction models. These interaction models are an attempted enhancement of the conceptual model. This is in order to more precisely define the input and output of each activity. Bustard and Lundy, 1996 argue that for this stage, an optional formal modelling technique is appropriate here. This is to facilitate consistency checks. The interaction models act as a bridge between conceptual models and DFDs or object models. They also support formal process oriented models. Formal modelling is achieved using a process oriented formal description language known as LOTOS.

MoIST versus RACE

Both the MoIST and RACE methods have a generally similar overall aim. This aim is to improve the requirements engineering process. Each of these methods however use varying approaches to achieving this aim.

RACE uses a formal modelling description language – LOTOS to define its generated interaction models. One drawback of this approach is that LOTOS is not well known in commercial software development environments. This has the consequence of significantly diminishing the usability and portability of the RACE method. RACE appears to be quite a good and workable method. However it still needs to do more work on using a linkage medium that developers will be more familiar with other than the formal modelling description language - LOTOS.

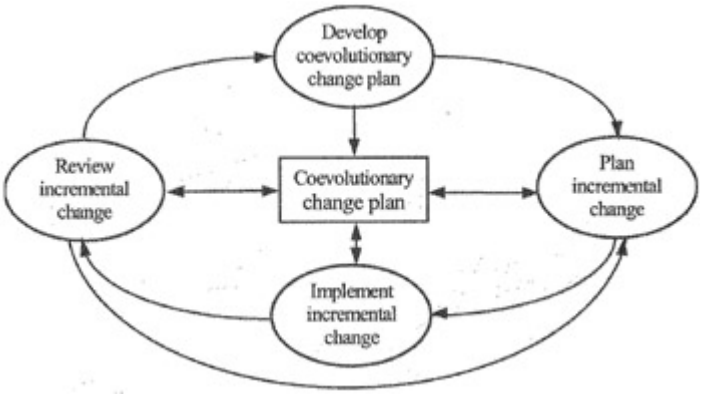
MOIST proposes to rectify the RACE method linkage problem by using UML-based techniques to provide the linkage with SSM instead of LOTOS. Software developers are more familiar with UML-based techniques than LOTOS and UML is more widely used commercially. This helps to ensure that it is more likely to be selected as a development method.

Business and Computing Support coEvolution (BASE) Method – Bustard, He and Wilkie – 2000.

BASE is a co-evolutionary framework for linked business and computing change. The motivation for this work was a desire to improve software engineering practice.

This was especially in the area of managing system requirements, Bustard et al, 2000.

BASE is goal oriented and goal driven. The BASE method is underpinned by an underlying coevolutionary change process. See fig below.



BASE co-evolutionary change process

The figure above describes the basic evolutionary change process in high level terminology. The co-evolutionary plan is created initially. This then acts as a guide to subsequent incremental changes. Each change is planned and reviewed. This may prompt adjustments to the co-evolutionary plan. Occasionally following the review, it may be recognized that the nature of the business or its computing support need to change substantially. This then prompts the creation of a completely new evolutionary plan. An example of where such a major adjustment might be required is in organizations switching to e-commerce as main model of customer interaction.

(Bustard, et al, 2000).

The first step in the Basic co-evolutionary change process is to 'Develop a co-evolutionary change plan'. This step is elaborated as four stages of activity.

I. Understand the situation of concern
II. Define the target system. This is the vision for the organization, in business and supporting IT terms. See fig 4.3.
III. Define the initial system. This is a description of the current way of working.
IV. Develop recommendations for change.

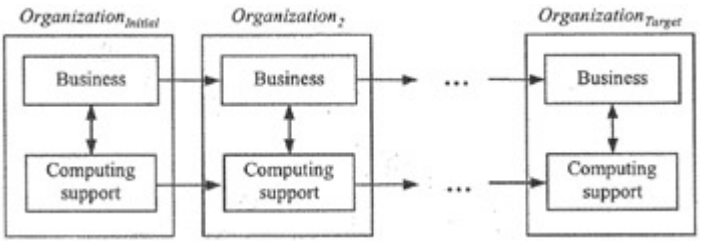


Fig 4.3 : Co-evolutionary change framework

The diagram in fig 4.3 describes the BASE framework model for change. Each business-computing support pair explains the state of the organization's point in its evolution. An organizational change may involve an adjustment to the business, its computing support or both. Each change is expected to retain or improve the business-IT alignment. The overall description if a co-evolutionary development plan, modelling how an organization might evolve towards a defined target state, through a sequence of several intermediate states (Bustard et al, 2000).

MoIST versus BASE

The BASE method bears some similarity to the MoIST method used in this research as they both define methods to link SSM to 'hard' information systems development. This aim however is achieved in different ways by each method.

The first significant variation is that the core architecture of MoIST and RACE methods and the means by which linkage is provided between SSM and hard systems development are dissimilar. BASE offers only one core option of achieving this linkage. The problem with this is that it does not necessarily provide software developers with the flexibility needed to maximise successful development. MoIST intends to address the problem and rectify it by providing analysis and development options depending on the assessed characteristics of each I S project. MoIST's multiple option method offers flexibility and choice to the developer that could help maximize opportunities for a successful project.

Yet another variation between MoIST and RACE is that each uses a different 'SSM culture'. There are two major proponents of SSM. These are the initiator of the culture Peter Checkland and his less known colleague Brian Wilson. (Checkland, 1981, 1990, Wilson, 2002 Bustard, 2000). MoIST is primarily influenced by Checkland's variation of SSM while BASE is particularly influenced by Wilson's approach to SSM. One potential problem with RACE's use of Wilson's "brand" of SSM is that not as many SSM methodologists are as familiar with the Wilson brand of SSM. Checkland's SSM on the other hand is more ubiquitous. (Checkland, 1981 etc etc, Wilson, 1981, 2002). This means that the RACE method could possibly be more utilised if the Checkland brand of SSM were used. MoIST intends to rectify this omission by utilising the Checkland brand of SSM that SSM methodologists are more familiar with globally. This should help in some way to optimise the instances of its use because of its ubiquity.

Business Object-Oriented Specification Technique – BOOST – Dobbin, Bustard, 1997

The main aim of BOOST is to integrate the business and computing analysis phases so that it is possible to move smoothly between them. This was one of the concerns that (Miles, 1992) had when he debated the merits of grafting versus embedding. It is assumed here that the BOOST technique offers one solution to the problem. (Stowell, 2002) did not quite agree that grafting or embedding posed an adequate solution. He offered an alternative known as 'navigating' instead of grafting or embedding. The technique offers one means of linking SSM and OOA, specifically to Shlaer-Mellor OOSA. Business analysis in BOOST is defined as a four-step process. Most of the analysis is standard SSM. Step three however is not an SSM activity.

1. Investigate problem situation
2. build activity models
3. refine activity models
4. make recommendations for change

The next stage of the BOOST technique produces Object Information models. This is a five stage development process. This process produces information models from interaction models. Transformation rules help to simplify the process. The five stages are:

1. Extract base objects
2. Extract base relationships
3. Identify additional relationships between base objects
4. Define the relationships between base objects
5. Refine the information model

MoIST versus BOOST

BOOST has been developed as an approach to linking successive phases of development. This technique allows for the products of one phase to be built directly on those of the preceding phase. This implies that an underlying linkage exists between the product sets, so that a change to one highlights or generates changes in the other.

One drawback to BOOST is that it offers one set way of linking the 'soft' phase to the 'hard system' phase. MoIST intends to rectify this drawback by offering a framework with three (3) different options that provide more flexibility to the developer. This can help the software development team to more accurately choose the development option suited to the software project.

Linking SSM to Structured Design Methods

(Stowell, 1995) was among the first persons to highlight the idea of linking SSM to existing structured design methods. He proposed that an agreed conceptual model could be expanded into a detailed data specification using a data flow diagram (DFD) (Stowell, 1995). Many other notable persons in the field have also had invaluable achievements in the area. Among these are (Avison and Wood-Harper, 1990) with Multiview and (Miles, 1988, 1992), who stirred up the grafting versus embedding debate. Work done includes Dynamic Systems Development (DSDM). This extends the RAD formally introduced by James Martin, 1991, (Avison and Fitzgerald, 1995) and Client led design by (Stowell and West, 1994).

Navigating the gap between action and a serving system: Champion, Stowell-2002

The authors are critical of attempts to form a bridge between systems thinking and hard systems engineering. Instead they propose to navigate certain gaps in the development process. This starts from the moment of inquiry within the organizational setting through to travelling to producing the artefacts for the logical development of the system. They highlight the distinction made by (Checkland and Scholes, 1990) about an information system being one that serves purposeful action. One criticism of prior attempts to move from conceptual models to logical design using data-flow diagrams (DFDs) was the abrupt change from conceptualising action to conceptualising data (Stowell, 2000, Mingers, 1995, Doyle and Wood, 1991). The method proposes to employ intellectual devices within the navigational phase. These are intended to maintain the sense of coherence from the ideas for action through to the serving activities. The concept of 'navigating' the gap is a means of creating a route from ideas for action to the requirements for an information system to serve the action (Champion et al, 2002).

Zachman Framework Integrating Business Process Models with UML Systems Models - 2001

This framework proposes UML as a means of modelling Business Processes. The problem that emerges with this that UML is originally oriented towards representing OO concepts. It therefore must be vastly extended in order to accommodate business modelling. As an OO system description notation, the UML is generally used to describe implementation views. It is also argued that using UML to describe the text oriented contextual and the conceptual view takes UML out of its existing domain and requires a mapping of the existing symbol to different concepts. The

alternative offered is to use a different notation for the higher levels process descriptions and use UML for the logical, physical and implementation views.

'Soft' Systems thinking and information systems: a framework for client-led design. Stowell and West - 1994

Here Stowell and West make a case for the use of client-led design as an answer to improving the efficiency of the traditional software development process.

Client led design is described as being a process where the client is handed the analysis tools and made responsible for finding out the problems and difficulties and bottlenecks in his or her own organisation. The Computing analyst acts as a guide to the whole process. The thinking here is that the client is most au fait with his own situation and is already intimately involved. This is a departure from the existing status quo where the analyst is the one with all the answers and expertise and spends days finding out and gleaning information. Stowell and West in order to facilitate the client led design provide a framework for the analyst relegated to 'guide' to follow. SSM is touted as the methodology of choice for the client to use in the finding out process. The authors insist that for SSM to be successful, it must be used as if the only intent were to find out and not to find out with the express aim of formulating a technical specification that will lead to implementation. They say it works better when their focus is solely on finding out. Stowell et al compare their client led design method to similar existing work done in the same field involving clients namely (Mumford, Mansell et al, 1990). They claim however that the identifying unique factor in theirs is that these approaches are undermined by the primary desire to fulfil a technological outcome.

This allows the 'client' or 'user' to have a greater control over the identification, specification and development of their information systems'. The traditional roles of Computer Systems Analyst (CSA) has been broadened by the authors to be Information Systems Analyst (ISA). Those who are most able to identify and discuss the implications of the information system are those who are one way or another, involved in its operation.

Miles' grafting versus embedding approach -1988, 1992

Conversations, ideas, debates and counter debates abound as to the efficacy of linking soft systems thinking with 'hard' information systems. Many theories have been put forward and many interventions have been made using some of these combined ideas. Nevertheless the area still has its proponents and detractors and

the great debate carries on. Some wonder if it is advisable or wise to transform a SSM conceptual model into a more hard systems oriented or more familiar DFD. (Miles, 1992) identifies the use of two techniques that he labels grafting and embedding. According to Miles, these two techniques are used when combining hard and soft systems paradigms. The grafting technique applies SSM to the problem and the outcomes and activities from the SSM are fed into the IS paradigm or they are grafted onto the hard systems engineering model. This technique while good is limited in its effectiveness by its lack of differentiation as evidenced in its failure to draw a clear distinction between object and information system as outlined by (Miles, 1992). This means that using the data model derived as output from SSM and using it as a front end to an object model type in the information systems domain could lead to conflict between model types. Another drawback to grafting is that usually once the process moves from analysis using SSM to design in the systems engineering domain. SSM is usually no longer utilised. In grafting, information systems utilises SSM concepts and products or deliverables.

Embedding on the other hand may be defined as a technique that incorporates into SSM the IS methodology or incorporates the information system hard systems domain into the Systems thinking or soft domain. Information system modelling demands both process and data analysis techniques but the conventional form of SSM is process oriented. Therefore for the embedding approach to succeed, it must incorporate into the methodological framework a means of modelling the data structure of an information system (Miles, 1992). SSM as it currently stands is process oriented. Information system models are data oriented. Information system modelling requires both process and data analysis techniques. For the embedding approach to succeed, it must incorporate into the methodological framework a means of modelling the data structure of an information system. Miles acknowledges this and accomplishes this by extending what he calls “SSM’s predication path” beyond the relevant systems and conceptual modelling stages.

Stage	Predication Focus	Outcome
3	What is the system?	Root Definition
4	What does the system have to do in order to be what it is?	Conceptual Activity model
4’	What are the information flows that will enable the system to do what it has to do?	Conceptual Flow Model
4’	What are the entity types?	Conceptual Data Model

Table 4.2: SSM's extended predication path table, adapted from Miles, 1992

The above table shows the extension to SSM. It renames the original conceptual model as conceptual activity model. There is then an intermediary 'conceptual flow model' and finally the generation or construction of the conceptual data model.

Information systems require both process and data analysis and design techniques. SSM is process oriented in nature. Miles proposed an SSM and ISD linkage in an earlier paper. (Miles, 1988). This stirred up many views and counter views. Miles expands and explains in a bit more detail on what his earlier, much debated approach was about. The grafting technique front ends the process oriented model type from SSM directly onto the data oriented IS. Miles is not in favour of this method as he argues that this would be a clash of model types (Miles, 1992). This view is supported by (Doyle and Wood, 1991). In the grafting approach, once the SSM part of it is completed, no more SSM would be carried out. The Miles embedding approach on the other hand encourages SSM to be continued throughout the entire software development process if needed. Here Miles emphasizes that it is not merely a matter of applying SSM in its currently existing form to the IS process. Consequently Miles proposes an approach that seeks to model the data structure of an information system. He poses a solution to extend SSM to include Checkland's conceptual models seen from three (3) different views (Checkland, 1981). The original conceptual model is now called 'conceptual activity model'. The information flows to the system are known as conceptual flow model. The entity type that supports the information flows are then defined and becomes the conceptual model.

Checkland	**Miles** (expands Checkland's conceptual model)
Original conceptual model	Conceptual Activity model
	Conceptual Flow model
	Conceptual Data model

Table 4.3: Three views of Miles' expansion of Checkland's Conceptual model

The whole aim of the embedding approach is to find some sort of data model to complement the process oriented SSM. Miles' aim is to have data oriented link to the data oriented hard systems engineering in order to avoid the clash of subjectivism and objectivism. The MoIST method builds on the embedding approach by facilitating a more seamless meeting of soft systems and hard systems. This is achieved by linking the Human activities in each conceptual model to the relevant activity diagrams in UML.

CCTA – 1993

This method provides a linkage between SSM and SSADM. The aim is to show how SSM can make better use of SSADM resources in problem definition, save time in subsequent requirements analysis and specification, and lead towards the development of applications which more fully satisfy business needs (CCTA, 1993).

In the resultant method, SSM is used to formulate Business System Options and SSADM to analyse current procedures and examine options for technical feasibility. The SSM used here is influenced by Wilson's approach to SSM especially in the usage of the Maltese cross. SSM is carried out first by doing the following:

1. rich picture building for initial scene setting
2. ensure business requirements and constraints are identified and retained during requirements definition
3. identification of information categories and production of information activity tables for a detailed understanding of the total information needs
4. identify scope and key activities for use in a Data Flow Model (DFD)
5. ensure all major interface points are identified
6. Maltese cross development and analysis for comparing required and existing information processing procedures and forming recommendations.
7. assess which aspects of the current situation require more detailed study.

The SSADM portion of the linkage is used for

1. requirements definition
2. analysing existing procedures
3. defining data requirements
4. data flow modelling and logical data modelling for definition of required functionality and data.

MoIST versus CCTA 1993

Separate logical models of system activities are developed and validated to gain a more comprehensive insight into the business area under study. Activities from individual models are combined to form a single model which can accommodate the various perceptions. Options for the design and implementation of an effective, efficient system to meet the requirements of the organisation may then be formulated. MoIST and the CCTA framework have a similar goal of linking the 'soft' computing paradigm namely SSM to the 'hard systems' paradigm. This is an effective method. The main deficiency is that SSADM is no longer the 'hard systems' method of choice for software developers in industry. MoIST proposes a solution to this drawback by the use of a link to a UML based context instead of SSADM. Another drawback was CCTA's use of Brian Wilson's brand of SSM which is not as widely known as Checkland's brand of SSM. MoIST's solution to this is to use the Checkland SSM instead in order to promote wider use of the method.

Checkland's COT Framework approach - 1993

Iterations of the process of SSM produce models which are widely agreed to be relevant in a company situation, then such consensus activity models can be converted into information flow models and the more traditional methods of information system design can be initiated. This provides the transition from the activity focussed SSM into the information – focussed outcome (Checkland, 1981). This means that the traditional 'project approach' to systems development embodied in and epitomised by hard systems engineering can now allow for a more holistic "process approach" where organizational flows, tasks and processes are taken into consideration. As with all problems of information provision, the first stages are nothing at all to do with data, hardware or software. They concern perceptions and politics, the interpretations of their world by the organizations in question (Checkland, 1988). Activity models offer a coherent basis for defining information related activities. These activity models are transformed into information flow models by asking about the information required, its form, the information source, the frequency required and the information generated. Usually at this stage, it is an appropriate time to discuss potential computer architectures. There is a general problem with introducing IT support within organizations in a coherent manner in order to carry out the purposeful activity of the organisation. The solution to this lies in conceptually

linking three domains ie conceptual, organisational and technology in what they refer to as a 'COT' framework (Checkland et al, 1993).

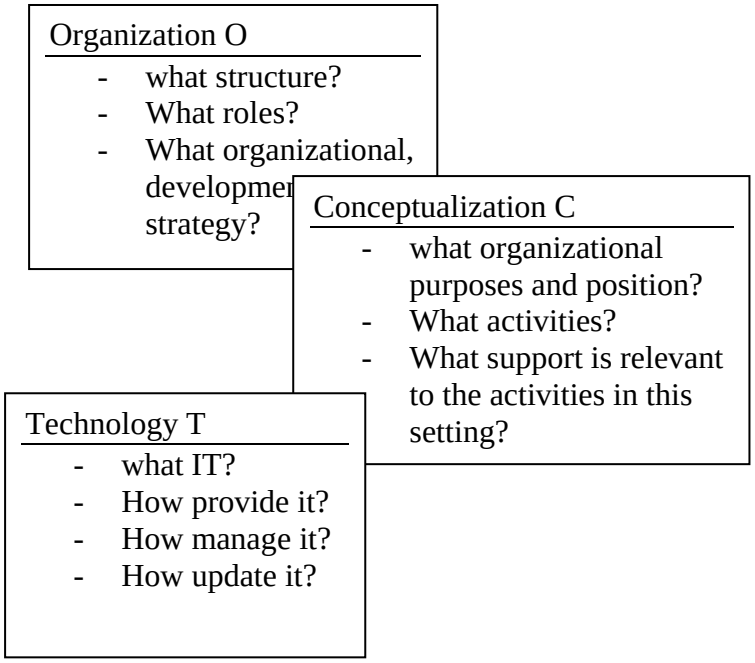


Figure 4.1: COT Framework, Checkland et al 1993

The technology domain ***T*** and the organization domain ***O*** stipulate and explore how Information technology can be more effectively utilised and introduced within the organization. The problem that usually arises when only these two domains are considered is that there is no explicit or coherent thought about how the two can flow harmoniously. This is where the conceptualization domain ***C*** comes in.

Domains ***T*** and ***O*** are thus connected by domain ***C***. It is referred to as the domain of explicit organised thinking about ***O*** and ***T*** where coherent thinking about ***O***'s nature and structure and IT support is carried out.

Domain ***C*** is perceived by the authors to be the crucial domain. As declared 'neglect of ***C*** as a conscious, organised, explicit activity is what leads to a premature leap from general statements of purpose to specific discussion of particular IT solutions'. SSM is touted as one of the ways of executing domain ***C***.

Dynamic Systems Development Method (DSDM) - 1991

DSDM is an extension of the Rapid Application Development (RAD). RAD was first formally presented by James Martin in 1991, (Avison et al, 1995). RAD evolved in

the 1990s out of the need for faster systems development than the traditional methodologies offered. RAD is an iterative, incremental approach. It compresses the traditional software development phases into shorter iterative cycles. It involves a small team of developers who work to tight deadlines to speed up the development process. In 1994 a consortium was formed to develop a framework that combined the best facets of existing methodologies with the development experience of RAD over the years. In 1994, a DSDM consortium was formed to establish nine (9) fundamental principles of RAD if it is to be used within the public domain. These principles retain the essence of the original features. It is additionally extended to address some of the evolving management, cultural and human issues that impact heavily on systems development environments that inform the debate. (Berger, Beynon-Davies, Cleary, 2004).

Multiview

Multiview originated as a response to traditional IS development methods that had strong roots in engineering discipline and technical rationality. (Vidgen, 2002). Multiview is structured in three tiers: general framework, local methodology and methods/techniques (collectively these constitute Multiview. Multiview now has two official versions. Avison and Fitzgerald's observation was that mid 90s onward saw the 'era' of methodology assessment. The original Multiview I – 1990 is a five (5) stage methodology. It incorporates SSM, ETHICS and other structured (hard) approaches pick the right ones for the problem situation. Multiview II – 1998 is more of a framework. This requires 'mediation' amongst four (4) components: organisational analysis, socio-technical analysis and design, information modelling and technical design and construction.

MoIST versus Multiview

MoIST and Multiview approaches bear some measure of similarities. This in terms of linking 'soft' and 'hard' paradigms. Multiview however incorporates ETHICS as well as SSM in order to link to SSADM. One drawback of Multiview is that though it works, there is no detailed step by step instruction as to how to traverse from SSM and ETHICS into the hard paradigm. MoIST intends to rectify this deficiency by providing a step by step instruction as to how to link both the 'hard' and 'soft' paradigms.

ETHICS

The ETHICS (Effective Technical and Human Implementation of Computer-based Systems) was developed by Dr Enid Mumford as a guide to user involvement in system design. The ETHICS method is intended as a guide to achieve a better balance between technology and people in the design of systems. (Mumford and Weir, 1979). In particular, the method advocates user involvement and participation throughout the design stage to produce a 'sociotechnical system' which will benefit both the business and the working environment of the users.

ETHICS is a method to help a design group (made up of management, users and technical experts) diagnose and formulate the problem, set objectives and develop alternatives, and take other appropriate actions right through to implementation and evaluation of the new system. Throughout development, emphasis is placed on both the human or social and the technical aspects of the system. Users develop social alternatives to improve job satisfaction.

Davis' Contingency Framework

The Contingency framework is useful for determining the situation in which it is best to use a particular method or approach. The framework is based on two variables, both concerned with different aspects of uncertainty inherent in the situation. These variables are requirements uncertainty and process uncertainty. The variables may be combined to give four types of organizational situations. We may use this to assist in determining the method most suited to the situation to produce an information system.

Davis(1982) describes how a multidimensional set of factors concerning the uncertainty of the desired system, type of users and type of designers may be evaluated and used to select the most appropriate requirements determination strategy.

MoIST versus Davis' Contingency Framework

MoIST has been influenced by the Contingency Framework in that the framework us useful in determining in which situation it is best to use a particular method. It examines two variables. These are requirements uncertainty and process uncertainty and determines which of the four options is best to use. The problem however is that once the option is selected, the contingency framework does not

specify how to achieve the desired result. MoIST however is very different in that it has two separate variables and three different options for software environment comparison. MoIST seeks to provide a solution for this potential drawback by detailing in a step-by-step manner exactly what is to be done to achieve each of the options chosen. MoIST will show how to achieve the suggested link from SSM to a hard systems paradigm. This is different from the contingency framework whch only suggests the linkage and not detail how to actually do the linking.

4.5 Real Life examples of Successful integration of Systems Thinking with IS paradigm

One of the motivations for developing a theoretical basis for combining methodologies was the fact that this was already happening in practice. A survey into the practical usage of SSM unexpectedly found that a wide range of methods were being routinely combined with SSM (Munro et al, 2002). The same tendency was noted among practitioners and they too see it as a justification for the need to look more closely at the whole context of combining methods (Mingers and Brocklesby, 1996).

Systems science combined with Object orientation were used together to increase the effectiveness of organizational requirements analysis for IS development. This was applied in a real-world case at the Labour Department (LD) of Hong Kong. A requirements specification was generated at the end of the project and delivered to the LD for subsequent design and construction of technological-based information systems (Lai 2000). SSM was used to provide learning in the situation that existed in the LD. It showed how the LD functioned amidst labour issues such as record high unemployment, closure of business operations and changes in legal standards. From an understanding of the situation, relevant systems of purposeful activities were formulated. An integrated framework method was then applied to the SSM findings. This led to the determination of information needs to support the defined purposeful activities. The outcomes here were then used to consider the data and technology that could yield the required information. At the end the intervention was an improved user requirements definition. This formed the basis of a successfully implemented system.

Yet another successful intervention that used combined methodologies was applied at Sainsburys. Sainsbury's is a leading UK food and grocery retailer. They are also owner of Shaw's and part owner of Giant Food in the USA. They have a profit of more than $1 billion and a turnover of about $15 billion. Sainsbury's was searching for ways to sustain its perceived technology lead into the 1990s. It also sought to maintain its outstanding record of yearly profit growth. In response to this challenge, a task force was formed to plan for the future. To support the strategy development process, intensive use of 'soft' OR and other systems methods were used. These included Soft Systems Methodology, cognitive mapping and strategic choice. It is believed to be the first time that these non-traditional approaches have been used together in one exercise for commercial purposes. The strategy that resulted from this process was implemented in 1995. Tangible and intangible benefits have begun to accrue such as cash flow savings, fewer stock losses and improved customer service. Tangible benefits include: a 5% increase in availability; a 10% reduction in stocks and 105 fewer losses from a branch stock control and ordering system; 30,000 additional stocking units from a new system for range control and depot stock reductions from a new purchase order system (Ormerod, 1995).

Benefits of combining Systems Thinking with Object Orientation

Real world problems are highly complex and multidimensional. Different paradigms focus attention on different aspects of the situation so the practice of combining methods provides the facility to deal effectively with the richness of the real world. Another significant advantage is that an intervention is not usually a single, discrete event, but goes through several phases. Methodologies tend to be more useful in some phase than others, so it is usually more effective to combine the methods in order to maximise its efficiency at every phase. This will help to achieve a more comprehensive level of success.

Limitations of combining SS with IS

The main challenge in combining SSM and Object Orientation is linking research methods together across different research paradigms (Mingers, 1997). At the philosophical level the issue of paradigm incommensurability exists. The attempt to provide guidelines for actually combining methods in practice has been fraught with difficulties. The alternative is for software practitioners who combine methods in

development to learn to live with and manage a degree of paradigm incompatibility (Jackson, 1997).

Other limitations exist at a cultural and psychological level. The cultural level is limited by the extent to which combining methods of diverse paradigms can be facilitated within organizations. The psychological challenge is the ease with which it is possible for a practitioner to move unhindered from one paradigm to another.

4.6 Conclusion

The list of successful linkages between Systems science and information systems development documented in this research is certainly not exhaustive. The concern in the systems community is that soft and hard systems approaches do not easily mix. If an SSM front end is grafted onto a hard systems development process there are three issues. If the same people do both the soft and hard phase can they do justice to both paradigms? The questions continue as the successes and failures occur. The limitations to mixing methods lie in the competence of the consultant and the participants rather than in the methods themselves (Mingers, 1997).

The particular method used for linking Systems Science and information systems is the Method of incorporating systems thinking into information systems design (MoIST). The MoIST is an amalgamation of the SSM and UML based method or process.

Chapter 5- Method of incorporating Systems Thinking into Information Systems design (MoIST)

'Methods are extremely important. They are prescriptive routes through the jungle. Follow a method that someone else has defined successfully, and you are less likely to be eaten by a tiger or to get lost in the undergrowth.' (Lunn, 2003, p 427)

5.1 Introduction

One of the benefits of integrating systems thinking in information systems development is the increased flexibility it provides. It pulls together two diverse paradigms. Their combined strengths provide a synergistic versatility and flexibility required for the unique nuances and facets encountered in software development.

Existing approaches to Requirements Engineering based on traditional Software Development models tend to emphasise technical knowledge, and are based largely on notations and prescribed processes (Checkland and Holwell, 1998, p xiii). Problem-solving needs a rich background of knowledge and intuition to operate effectively. Breadth of experience is also necessary so that similarities and differences with past strategies are used to deal with new situations (Bubenko, 1995).

Traditional methods of requirements determination assume that requirements can be successfully obtained with no knowledge of the organization. However this assumption may produce a poor quality system, as methods address the wrong problem by ignoring organizational knowledge that is part of the requirements. It is assumed that the user knows best, and that the developer does not need expert domain knowledge. But it is becoming clearer that knowledge of this type does help in determining requirements (Flynn, 1998, p 141).

Companies have come to recognize that traditional software development practices are inadequate from both a technical perspective and a business perspective. This is causing companies to reengineer their software development or acquisition processes (Jacobson, 2000, p 4). The requirements process is arguably the most important process within systems development as studies have shown that the majority of errors are made here. There is increasing attention being paid to social rather than to technical factors in the process (Flynn, 1998). For software developers there is a widening gap between the degree of flexibility and creativity needed to adapt to a changing world and the capacity to do so. They view the difficulties as attributed to individuals or groups not willing to engage in effective and efficient

processes of innovative design (Thomas et al, 2002). Developers typically fail to spend sufficient time in the early stages of design: problem finding and problem formulation. Subsequently they often then bring critical judgment into play too early in the idea generation phase of problem solving (Flynn, 1998).

The essence of this research is the design of the MoIST method for integrating systems thinking into information systems design of the entire software development process, but in particular the requirements elicitation phase. This integration ensures a greater incidence of project development being completed successfully and in a timely manner (Bustard, D, Kawalek, P, Norris, M, 2000, chap 13).

A method is a way of doing things and methodology is the study of ways of doing things (Lunn, 2003, p 427). Additionally, a method is a planned procedure by which a specified goal is approached step by step, not to be confused with methodology, which is the science of methods (Jacobson et al, 1992, p 30). A basic requirement of a good method is that it simplifies the development of systems that have the software architecture it is meant for (Jacobson, 2000, p 44). The combination of these two epistemologies in the new method formation simplifies the process as it allows both strengths to be maximised and weaknesses minimised (Bustard et al, 2000).

5.2 The MoIST Method

Is there any method that is not based on another method? My early object-oriented design method developed in 1967 was heavily based on a design method (at least ten years old at that time) used within Ericsson to design telecommunication systems. You can take any other method and you will find similar relationships to earlier works. One of the most popular methods, OMT, is probably an even better example of a method based on methods. The uniqueness of that method is how it has been composed of other people's work. *To further develop methods based on other methods is most natural and that will continue to happen* (Jacobson, 2000, p 165).

MoIST stands for Method of Integrating Systems Thinking within Information Systems design. It is a method that has been developed out of research on various IS methods and modelling approaches (Davis, G, 1982, Avison and Wood-Harper, 1990, CCTA, 1993, Bustard, 1997). It also combines the social and human factors of SSM with the more technical UML-based method. MoIST has been influenced by the Contingency Framework of (Davis, G.B., 1982). It is also influenced by the work done

by the government centre for information systems (CCTA, 1993) in combining SSM and SSADM and by Multiview Method by (Avison and and Wood-Harper, 1990).
The contingency framework of (Davis, 1982) prescribes a soft or hard approach for each software development project based on the level of uncertainty about what process to use and the level of uncertainty about what the requirements for the project are. Davis's framework describes how a multidimensional set of factors concerning the uncertainty of the desired system, type of users and type of designers may be evaluated and used to select the most appropriate requirements determination strategy(Davis, 1982). Multiview combines new methods with a soft approach by adding new stages and the required iterations (Wood-Harper, 1985) and the CCTA documentation shows the phases of SSADM and the possibilities for extension (CCTA, 1993).

Bodies of work related to MoIST and that combine SSM with UML already exist (Bustard et al, 1996). MoIST's major area of uniqueness lies in the juxtaposition of its three (3) development options and their level of replicability in organizational IT projects. MoIST also complements the existing body of related work and adds to the arsenal of successful software development methods; as it facilitates projects that match its specific characteristics (see chapter 5).

Creativity has been described as a balance of convergent and divergent thinking appropriate to the situation. This balance is essential in undertaking software development, which may be considered as a class of creative problem solving (Nickerson, 1999). The MoIST method combines discipline and creativity. Discipline and creativity are the *odd couple* of software development – the discipline imposed by methodology, for example, forms a frame for the opportunistic creativity of design. This provides a base that enables software developers to both create and engineer the systems they build: to be adaptable to the changing environment that is inevitable in the software development discipline (Glass, 1995).

5.3 The UML and the MoIST Method

The Unified Modelling Language (UML) advocates the innovation of new methods like MoIST. It provides extensibility mechanisms so that future modelling approaches can be grown on top of the UML (OMG, 2003).

As the strategic value of software increases for many companies, the industry looks for techniques to improve quality and reduce cost and time-to market. One commonly underused technique in the software industry is modelling. Developing models for a software system prior to its construction or renovation is as essential as having a blueprint for a building. Software system models help in the comprehension of such systems in their entirety. There are many factors of a project's success, but one essential factor is having a modelling standard. *The UML must and can support various methods and processes of building models. The UML can also support multiple development methods without excessive difficulty* (OMG, 2003).

This research does not assume that an information system will necessarily be the best solution to an apparent problem. It promotes the establishment primarily of the key aspects of organizational structure. This is then blended together with such influencing factors as environmental characteristics, technology and task. Organizational outcomes will usually depend on negotiation between the different key actors - organizational participants, as there will always be different solutions to problems or short-term versus long-term views (Hirscheim et al, 1995).

Software development has been described as a *'craft'*. The negative connotations of this label include an inability to consistently guarantee a quality product, fit for the purpose for which it was developed, produced on time and within budget (Standish, 1995). A study of over 8,000 projects (Armarego et al, 2002, Standish, 1995) reported 16.2% of software was successful, 52.7% were over budget, time and had fewer features and 31.1% of projects were cancelled. These rates do not significantly differ from those reported in the 1970s and 1980s. Many of the shortfalls may be traced to deficiencies in formulating a description of the system to be developed (Mann, 1996).

Deciding on which systems should be built remains problematic and is an analytic challenge ill-served by current methodologies. Such theoretical limitations become more pronounced when we recognise that these methodologies hardly begin to address potential research issues in organizations, for example privacy and data integrity (Winter et al, 1995).

This MoIST method attempts to redress these deficiencies. It gives developers the opportunity to choose the best option within the method. They are enabled to assess the characteristics of their project and determine the best way of development. This increases the likelihood of success. All good methods support object modelling (Jacobson, 2000, p60).

One of the benefits of the MoIST Method is its flexibility. It integrates two diverse; but flexible methods. This increases the total flexibility of the integrated method. This versatility and flexibility is needed for software development as there are various nuances and unique facets of each project that might not fit exactly into every option in the method. No project is likely to follow the UML to the letter. Rather, the aim is to select out the parts of the process that are relevant to the current project and organisation (Lunn, 2003, p 429)

The MoIST method suggests heuristic solutions to problems that are known to be hard to characterize. Having several project options within a method is integral to most research domains (Jacobson, 2000). The MoIST Method is designed for both users and developers to work together to maximise their strengths in order to improve the requirements elicitation process. The success or failure of the requirements elicitation process usually determines the outcome of the entire software development project (Flynn, 1998). Requirements are not objective, unchangeable artefacts, available at the start of the process, to be “captured” like butterflies. Instead, they are emergent and are *social constructions* resulting from interactions involving users and developers in the process (Dobson and Strens, 1994).

5.4 The MoIST Model

The MoIST model is an integral part of the MoIST method. The MoIST method combines elements of the MoIST model, ProcessMoIST, MoIST Project Selector Tool, the MoIST Project options and MetricsMoIST.

The MoIST method caters to development projects with differing degrees of structuredness. This is in terms of the levels of development of the MoIST method's core determinant variables. These are the ***requirements certainty*** and the ***development environment*** of each project.

The ***structuredness*** of the software development environment relates to the degree of knowledge we have about the problem to be solved. Therefore when requirements are not clear, we have a lower degree of knowledge about the problem domain. The degree of knowledge consequently dictates the structuredness of the environment which in turn influences the project option chosen.

The MoIST model has been developed to flexibly use one or more project options within the project lifecycle. The degree of structuredness catered for by the MoIST method range from very unstructured – Option A, fairly unstructured – Option B to very structured – Option C. In order to more fully redress the software development weaknesses identified in the research (pg 2 and Chap 1), MoIST tends towards providing a viable development route for ***unstructured*** development environments. In order to more accurately test MoIST's assertions, the most suitable case studies were from unstructured environments. Consequently in the research, there is no case study that explores the Option C pathway. The provision of Option C has been made however for any possible clients who still desire to utilise MoIST as a solution even though requirements certainty and the structuredness of their development environment might be very high.

Overview of MoIST Project Options A, B and C

Project Option A

MoIST is designed to always begin at project option A. In option A, the entire project is preceded by a comprehensive SSM study. Comprehensive here means starting with rich pictures and ending with a Conceptual Primary Task Model. This facilitates a thorough exploration of the project and its characteristics. It also enables the development of conceptual activities that might be needed for more advanced design. After project option A is completed, a project status appraisal is conducted using the MoIST project selection tool. This is a very important phase in the MoIST

process. The findings of the SSM study will determine whether or not it is feasible to continue with information systems design or whether a more relevant solution lies in a non-information systems alternative. If the findings point to discontinuing systems design, then a feasible alternative is pursued. This decision is quite acceptable and the MoIST method would have effectively prevented the needless waste of resources on progressing to a more advanced design. If the SSM findings show that it is viable to continue along the information systems design route, then an appropriate project option is chosen using the MoIST project option selector tool. The MoIST project option selector tool determines whether or not the development environment is currently structured or unstructured. The resultant degree of structuredness determines whether Option B or option C is chosen. If the environment is still considered to be unstructured, project option B is chosen. On the other hand, if the environment is considered to be very structured, project option C is chosen.

Project Option B

Option B facilitates the transition of the software project from the SSM study to the UML-based phase. This involves the derivation of conceptual models from each root definition, determination of selected activities as candidates for IT support and mapping of required services onto objects. If the environment is now considered to be structured, project option C is chosen.

Project Option C

In project option C, the transition is made from SSM Conceptual activities to UML-based activity diagrams. Project option C is chosen when the development environment is very highly structured. It is also used when the requirements are very well understood and can be applied in their current state as they are. Not many software engineering projects fit the above criteria, therefore it is not expected that MoIST's option C will be a popular option chosen for development. This is because most software projects are being developed in an unstructured environment. Based on the literature review in Chapter 1, it was seen that software projects are increasing in complexity and unstructuredness and that a greater understanding of human-social factors in Computing is needed to aid successful development. MoIST tends towards projects that are unstructured. Nevertheless, in order to accommodate any structured project that would want to utilise MoIST, Option C is offered as a route to

successful development. Project C is in essence a hard systems option that provides a link to using MoIST for any possible project that might have a very structured environment where requirements are fully known.

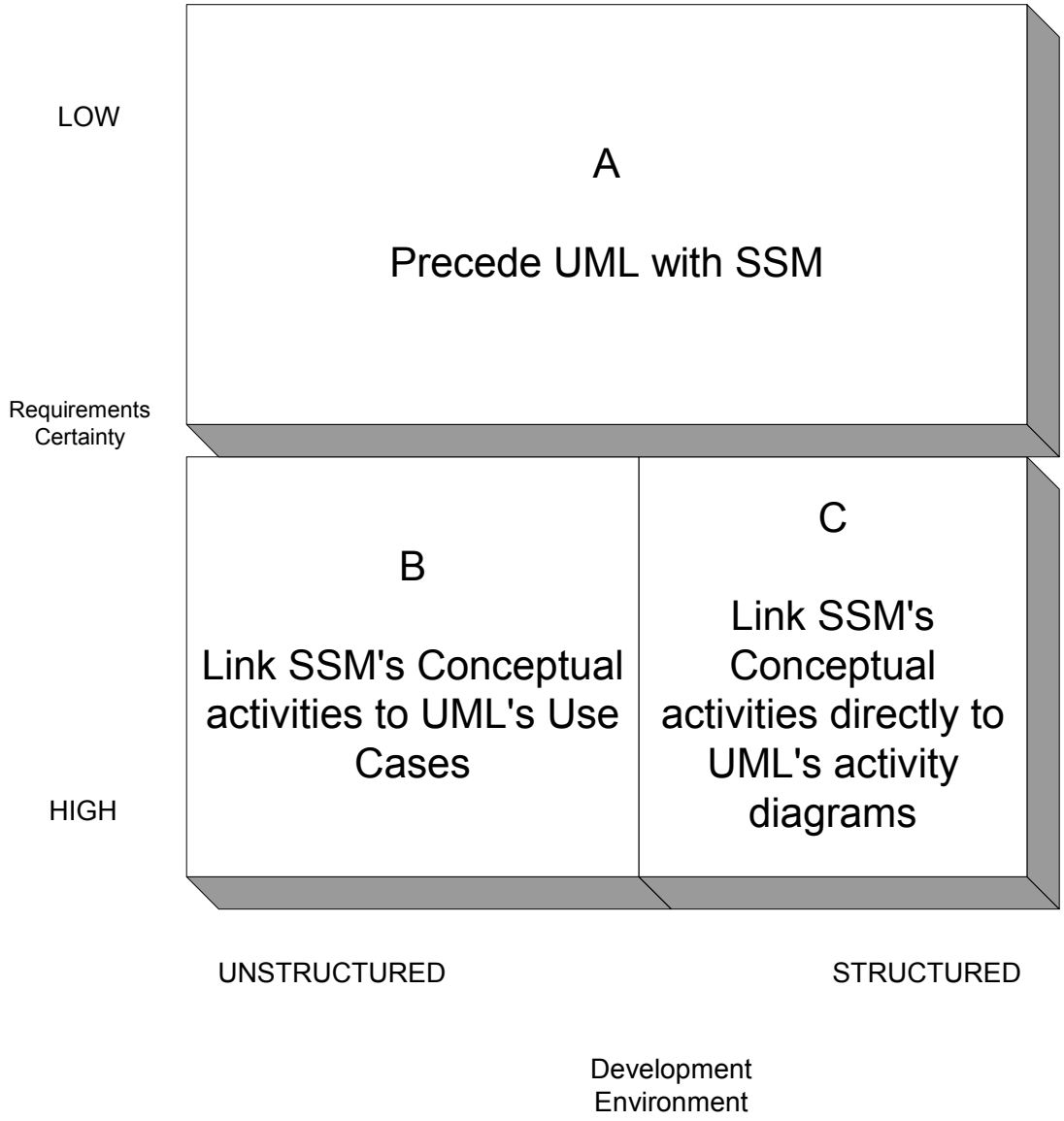


Figure 5.1: The MoIST Model

Based on studies of the different characteristics of the methods and their respective approaches, one possible conclusion is that methods have developed to suit different situations. For example, some methods, such as SSADM and participative systems design, assume that a current system always exists, which can be used as a basis

for analysis and design of new systems. Other methods, like the Checkland methodology, do not make this assumption (Flynn, D, 1998, p 351).

The MoIST method describes how a multidimensional set of factors concerning the uncertainty of the desired system, type of users and type of developers may be evaluated and used to select the most appropriate requirements determination strategy. MoIST shows how the specific software development project fits onto a planning grid which houses the MoIST method with its three (3) options.

The MoIST method is based on two variables, both concerned with different aspects of uncertainty inherent in the situation. These are requirements certainty and the nature of the development environment. This shows how the specific software development project fits onto a planning grid which houses the MoIST method with its three (3) options. A limitation of the simplified model above (see fig 5.1) is that there is a tendency for the status of the two variables to rely heavily on the judgement of the project manager. This can be mitigated by involving more stakeholders in the initial evaluation process.

Requirements Certainty: - This is the extent to which the requirements are known and fixed. The level of requirements uncertainty is ascertained by checking for certain characteristics in the project. This will determine whether requirements uncertainty is low or fixed. Low requirements uncertainty characteristics include conflicting interests among stakeholders and/or developers, likely organisational changes, the proposed system is contentious or where the proposed system crosses functional or organisational boundaries. The presence of one or more of the stipulated characteristics in a project makes it a likely candidate for 'low requirements certainty' status. For the requirements certainty level to be deemed 'high' the requirements should be very clear, with no ambiguity and all parties should agree.

Development Environment: - This is the predictor of the structuredness or unstructuredness of the software development environment. The level of structuredness or unstructuredness is ascertained by the general characteristics of the project. For example, developers not being able to agree will give the environment, 'unstructured' status. The environment being relatively contention free will give the project 'structured' status. The status is determined by the project management team.

These two variables may be combined to give three (3) types of organizational situations (see fig 5.1). This is used to facilitate determination of the option in the MoIST method most suited to the development situation. Where requirements certainty is low and development environment is structured or unstructured, the entire UML phase needs to be preceded by SSM. A project may have as it characteristics, requirements certainty being high and the development environment unstructured. Using the MoIST method means that UML-based method's initial stage is enhanced with SSM. SSM's conceptual activities can be linked to UML's activity diagrams when there is high requirements certainty, a structured development environment and the requirements are very clear to more than 90% of the development team. This option might be rarely used as unstructured project requirements are not usually that clear. Also linking conceptual activities to activity diagrams means bypassing use cases. This linkage lends itself to less accuracy than linking conceptual activities to use cases. This option was not explored in this research with a case study, but might act as a catalyst for other researchers as methods are inspired by and built on existing methods. (Jacobson, 2000). The characteristics of the project merely serve as a guideline. Owing to the variegated nature of most software projects, the developer is free to use the characteristics as a guideline to see which option the project slots into

5.5 Process MoIST (ProMoIST)

The presence of a well-defined and well-managed process is often a key discriminator between hyperproductive projects and unsuccessful ones. The UML is intentionally process independent and defining a standard process is not a goal of the UML or OMG. UML encourages various organizations to use the same UML diagram types in the context of different processes. The UML recognizes the importance of process. The reliance upon heroic programming is not a sustainable business practice. Processes by their very nature must be tailored to the organization, culture and problem domain at hand. (OMG, 2003).

Process MoIST is a well defined Process Framework that undergirds the MoIST method. The order inherent in Process MoIST enables the advancement of the project in an orderly and as stable a manner as is possible. Process MoIST lends structure to unstructured I S Development projects. It provides linear and iterative progression of a project from analysis/elicitation to design, implementation and evaluation. Whilst Process MoIST does not actually carry out implementation; its

effects on implementation are indirect. This is because it facilitates the successful analysis and design which can more easily extrapolate to successful implementation.

5.5.1 MoIST's Project Option Selection Tool (MoPros)

	25 points	25 points	25 points	25 points
Project Options	**Users**	**Developers' skillsets**	**Organizational environment**	**General characteristics**
A	Users are a bit unsettled as they are experiencing organizational changes	Requirements known at this point are relatively clear to 80% of the development team	Development environment unstructured	Proposed system is to replace or enhance an existing system
B	Users uncertain about the need for the proposed system or users opposed to the proposed system	Developers not able to agree about requirements	Development environment has pockets of structured and unstructuredness.	Conflicting interests and the proposed system might cross functional borders
C	Users open to the new system	Requirements known at this information are very clear to 90% of the development team	Development environment is quite structured	Environment is relatively contention free

Fig 2: MoIST's Project Option Selection Tool (MoPros)

MoIST Project Option Selection tool is a pre-defined template of the general characteristics of the average software development project. These characteristics are delineated into related project options. These characteristics were selected as the core characteristics after examining other IS project management literature (CCTA, 1993, Flynn, 1998, Davis, 1982, Avison and Wood-Harper, 1990). The MoPros presents three (3) project options. Each option defines its own user types, developers' skillsets, organizational environment and general characteristics. The project team manager is responsible for facilitating the selection of the most appropriate project option at any given point in the MoIST process. The selection of another individual project option B or C within the Process MoIST framework is dependent on assessment of the problem domain, implementation

technology and team skills sub-options within each project. The current project characteristics are then matched using the Project Option Selection Tool. A maximum of twenty-five (25) points are allocated to each sub-option. The project option with the highest total points or percentage score is automatically deemed to be the most appropriate project option. The scoring process works with the project team meeting to provide consensus on the scores. Each team member is given a copy of the MoPros template with the maximum score. As they discuss each option, each member is required to allocate a score out of 25 for each section. The scores for each option are averaged and the project option with the highest percentage determines the option path taken. In the event that the project manager feels that the scoring decision is inaccurate, the manager has the autonomy to take a managerial decision to change this score. This is because ultimately the responsibility for the success of the projects lies with the project manager.

There is no limit to the number of times that the MoIST Project option selection tool may be used within the duration of the project lifecycle. As progress is made, the project status is subject to change. If the project status changes, the current project characteristics can be reappraised. This reappraisal is done using the MoIST project Option Selection Tool. Status changes may become more evident as project deliverables are achieved. The selection tool is then used to choose a more appropriate option that reflects the project's updated status. One practical example of this happened during this research. Initially one project option was chosen using the MoIST project Selection Tool. The project activities were subsequently carried out. Over time, the successful outcomes of doing the activities precipitated the need for a new appraisal. The project was re-examined in the light of new developments. The MoIST project selection tool was used and a more suitable project option was then chosen (see Chap 6).

There is a distinct advantage in enabling the MoIST Project Selection tool use to be iterative within a project lifecycle. This is because it acts as a monitoring and corrective mechanism. This helps to ensure the integrity and accuracy of the MoIST method. This in turn ensures that the integrity and rate of advancement of the development project is not compromised.

How Process MoIST works

Process MoIST is an architecture-centric, SSM-UML driven, iterative and incremental process framework.

1. It provides guidance as to the order of a project team's activities
2. It directs the tasks of individual developers and the entire team
3. It evaluates the Project and assesses its perceived characteristics. This is done against the characteristics set out in the MoIST Project Option Selection Tool template.
4. It specifies what I S artefacts should be developed
5. It produces a statement of work that describes the system
6. It offers criteria for monitoring and measuring a project's products and activities.

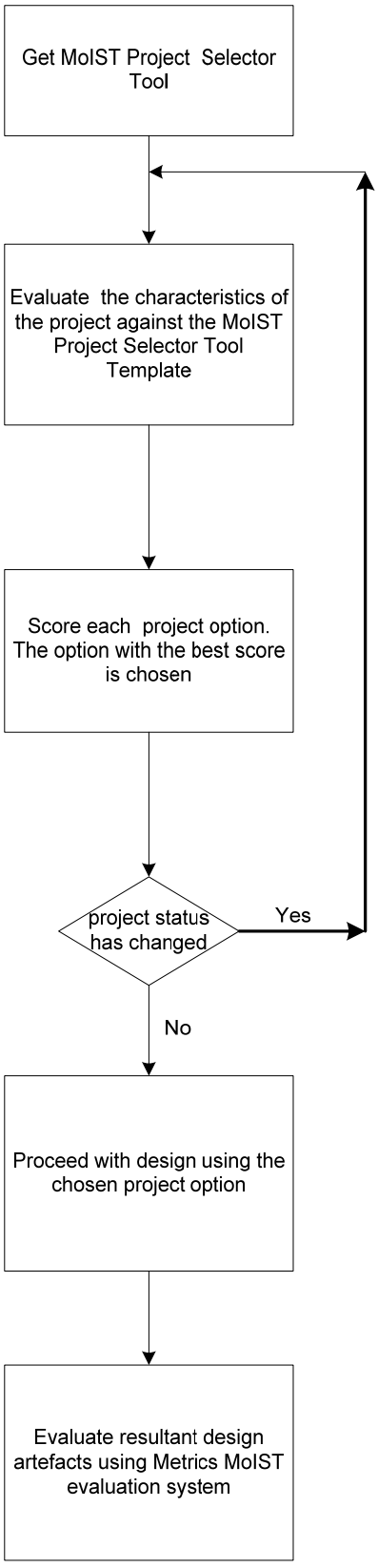


Fig 5.2: **Process MoIST Procedures**

5.6 The MoIST project options A, B and C are shown in more detail below.

MoIST Project Option A

Precede UML with SSM

Status: Requirements Certainty (Low) + Development Environment (Unstructured or Structured)

MoIST Option A's Activities
1. Requirements for computer-based information system 2. construct rich picture 3. develop relevant issue-based and primary task root definitions and conceptual models 4. derive consensus primary task model and information categories 5. formulate the recommendations for information system design

This option allows for the organization and domain to be studied and modelled. The results of this SSM Analysis can be fed directly into the initial phase of the UML. In this option, depending on the expertise of the SSM methodologist, they can start with any step thought necessary. They can begin with rich picture construction or conceptual models.

After finishing this design phase, the resultant artefacts are then evaluated using MetricsMoIST Evaluator System.

MoIST Project Option A's Construction framework

Option ID	Activity Label	Construction Procedure
A1	Gather the known requirements	1. Conduct informal interviews with stakeholders. This helps to observe the area under study in the context of the whole organisation. 2. Formulate and administer questionnaires. This considers multiple perceptions and recognise that some may conflict 3. Identify and assess the business requirements, organizational and cultural ethos and the financial and technical risks.
A2	Construct Rich Picture	1. Capture, describe and express the problem situation diagrammatically. The rich picture is unique to the analyst as it is the analyst's perception of the problem (see chap 2) 2. Ask questions such as, what roles or people have relationships with the situation described? And what organisational issues are there within the situation? 3. Illustrate all elements of the problem without being overly detailed. 4. Use meaningful symbols to represent relevant components of the situation and arrows to illustrate relationships.
A3	Develop primary task and issue based root definitions	1. Consider two (2) types of root definitions (RD). There are those related to the primary task of an organisation and those related to issues within the situation. (see chap 2) 2. Structure the root definitions according to CATWOE (see chap 2).
A4	Develop conceptual models	1. Derive conceptual models from the primary task root definition (RD). This defines what a system would have to do to be the system described in the RD. 2. Use verbs to describe the logically linked activities that satisfy the RD
A5	Derive consensus primary task model (CPTM)	1. Construct the CPTM from elementary primary task model 2. Assess the feasibility of each activity within each primary task model and assemble those acceptable to the stakeholders
A6	Derive information categories	1. Take each activity in the CPTM. 2. Derive the broad, distinct groups of information needed to support the activity along with the information categories generated by doing the activity.
A7	Make recommendations and	1. Produce a feasibility report to document decisions as to whether and how further work

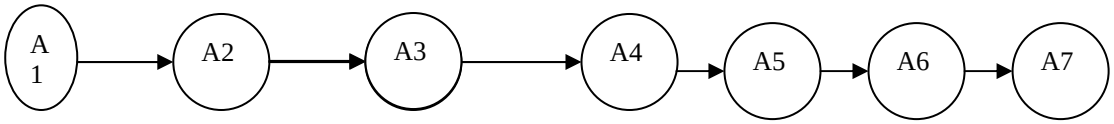
	then select Option ID A7.1 or A7.2 or A7.3	should proceed
A7.1	Use MoIST Project Option Tool selector to decide between option A or B	1. Use Option Tool selector to determine the current status of the project (see chap 5-MoIST method)
A7.2	Choose an alternative solution.	1. Proceed in a different direction from that first envisaged
A7.3	Suspend or cancel project	1. Suspend or cancel project as not deemed feasible

Option ID	Skillset	Deliverables	Monitoring	Active status (Retained/ Modified/ Dropped)	Supervisor
A1	SSM	Stakeholder requirements	Scheduled stakeholder feedback meetings	Retained	
A2	SSM	Rich Picture	MetricsMoIST		
A3	SSM	Root Definitions (RD)	MetricsMoIST	retained	
A4	SSM	Conceptual Models	MetricsMoIST + CATWOE	retained	
A5	SSM	Consensus Primary Task Model (CPTM)	MetricsMoIST	retained	
A6	SSM	Information Categories	MetricsMoIST	retained	
A7	SSM	Recommendations of solution	MetricsMoIST	retained	

Timeline for MoIST project option A

Activities	Completion time (Min Hours)	Completion time (Max. hours)	Error/Feedback & Correction Time (Hours)	Total time (hours) (max+error)
A1	20	25	7	32
A2	5	7	3	10
A3	12	13	4	17
A4	13	15	4	19
A5	6	7	2	9
A6	8	9	4	13
A7	6	8	3	11

Gantt Chart for Project option A



MoIST Project Option B

Identifying and defining use cases from conceptual activities.

Status: Requirements Certainty (High) + Development Environment (Unstructured)

MoIST Option B's Activities
B1. Derive conceptual primary task model (CPTM).
B2. Select and prioritise Conceptual model activities.
B3. Determine which activities require further decomposition.
B4. Determine which of the selected activities are candidates for IT support.
B5. Identify actors.
B6. Develop high-level use cases.
B7. Develop multi-level use cases.
B8. Identify high-level objects.
B9. Map required high level services onto objects.
B10. Continue design.

MoIST Project Option B's Construction framework

Option ID	Activity Label	Construction Procedure
B1	Derive Conceptual Primary Task Model (CPTM)	1. Derive each conceptual model from the root definition 2. The model should define what a system would have to do to conform to the root definition 3. Identify the logically linked activities that would have to take place to satisfy the root definition.
B2	Select and prioritise Conceptual model activities	1. Identify and place each activity into either high-level or low-level categories. 2. High-level activities are those which have the potential to be decomposed into more specific activities. 3. Low-Level activities are those which can be used immediately without further decomposition.
B3	Determine which activities require further decomposition	1. Choose the high-level conceptual activities already selected. (see B2). 2. These activities may require further decomposition of some specific activities for which use of IT is currently vague or unclear.
B4	Determine which of the selected activities are candidates for IT support	1. Select the low-level conceptual activities which require no further decomposition. These can be used immediately.
B5	Identify Actors	1. Identify the functional roles (person or system) associated with each low level activity
B6	Develop high-level Use Cases	1. Let each identified low-level activity serve as the name of the use case 2. Involve the relevant actors and domain experts when writing up these high level use cases
B7	Develop multi-level use cases	1. Decompose the high-level use cases to an appropriate number of levels 2. Continue to involve the relevant actors and domain experts in order to derive and validate the use cases
B8	Identify high-level objects	1. Identify the objects from the use cases by extracting the nouns 2. Remove duplicate objects 3. Form associations between the objects to show their relationships
B9	Map required high-level services onto objects	1. Determine the high-level services from the SSM conceptual Model Information Requirements 2. Map the services onto the objects

Option ID	Skillset	Deliverables	Monitoring	Active status (Retained/ Modified/ Dropped)	Supervisor
B1	SSM	Conceptual activities linked into CPTM	CATWOE to ensure conceptual activity integrity	Retained	
B2	SSM + UML	Conceptual activities prioritised and placed into either low-level or high-level categories	1. Further decomposition to be done = high-level. 2. No further decomposition possible = low-level.		
B3	SSM+UML	Candidates (if any) appropriate for IT support	MetricsMoIST	retained	
B4	UML	List of UML actors	MetricsMoIST	retained	
B5	UML	High-level use cases that further decomposed	Noun extraction process	retained	
B6	UML	Multi-level use cases	MetricsMoIST	retained	
B7	UML	High-level objects that can be further decomposed	MetricsMoIST	retained	
B8	UML	High-level services mapped to objects	MetricsMoIST	retained	
B9	To be determined by project team	Continuation of UML design activities	MetricsMoIST	Retained	

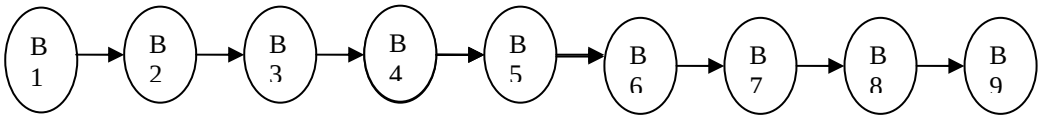
Timeline for MoIST project option B

(Min Hours) (Max. hours) (Hours)

(max+error)

Activities	Completion time	Completion time	Error/Feedback & Correction Time	Total time (hours)
B1	2	5	2	7
B2	1	4	3	7
B3	1	3	2	5
B4	2	3	1	4
B5	2	4	2	6
B6	2	5	1	6
B7	1	4	3	7
B8	2	3	2	5
B9	3	5	1	6

Gantt Chart for Project option B



Graph showing Project option B Activities vs. Total Completion Time

MoIST Project Option C

Link SSM's conceptual activities directly to UML's activity diagrams
Status: Requirements Certainty (High) + Development Environment (Structured)

MoIST Option C's Activities	
C1.	Derive the Conceptual Primary Task Model (CPTM)
C2.	Identify conceptual activities that are candidates for IT support
C3.	Identify the nouns from conceptual activities
C4.	Link conceptual activities directly to the UML activity diagrams

This method delivers a relatively simple solution by direct transition. It is quick to apply and requires no further investigation into the business domain as the resultant activity diagram model is based solely upon the CPTM and supporting root definitions. Every process needs to be evaluated and ProMoIST is no exception. The

MetricsMoIST Evaluator System developed in this research is used to determine the efficacy of the design artefacts developed in the research.

MoIST Project Option C's Construction framework

Option ID	Activity Label	Construction Procedure
C1	Derive Conceptual Primary Task Model (CPTM)	1. Derive each conceptual model from the root definition 2. The model should define what a system would have to do to conform to the root definition 3. Identify the logically linked activities that would have to take place to satisfy the root definition.
C2	Identify conceptual activities that are candidates for IT support.	1. Identify and select the activities within the CPTM that have the potential to be investigated for possible IT support. 2. Look for the conceptual activities that most closely match the key areas gleaned from the SSM study and that matches the client's needs.
C3	Identify nouns from Conceptual activities.	1. Select each activity in turn and identify the nouns in the CPTM activity bubbles. 2. Use the nouns selected in (1.) above to be the candidate activities for linkage to the UML activity diagrams. 3. Examine the list of nouns and remove any duplicates or any nouns which represent the same entity.
C4	Link conceptual activities directly to the UML activity diagrams.	1. Determine the logical dependencies and information requirements of each conceptual activity within the CPTM. 2. Form the relevant associations between the UML activity diagrams to form an initial linkage. 3. Use the initial linkages to form subsequent linkages that will accelerate the design process.

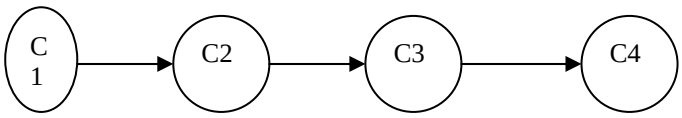
Option ID	Skillset	Deliverables	Monitoring	Active status (Retained/ Modified/ Dropped)	Supervisor
C1	SSM	Conceptual activities linked into CPTM	CATWOE to ensure conceptual activity integrity	Retained	
C2	SSM + UML	Conceptual activities that qualify for IT support	CATWOE + MetricsMoIST	retained	
C3	SSM+UML	Nouns derived from conceptual activities	CATWOE + MetricsMoIST	retained	
C4	SSM + UML	Linkages and associations between conceptual activities & UML activity diagrams	CATWOE + MetricsMoIST	retained	

Timeline for MoIST project option C

(Min Hours) (Max. hours) (Hours) (max+error)

Activities	Completion time	Completion time	Error/Feedback & Correction Time	Total time (hours)
C1	2	5	2	7
C2	1	4	3	7
C3	3	6	3	9
C4	5	8	4	12

Gantt Chart for Project option C



Graph showing Project option C Activities vs. Total Completion Time

5.7 MetricsMoIST Evaluator System

Metrics have been applied and misapplied to software systems for decades. Ideally, metrics provide a measure of quality and the location of defects in a system. A metric is a measurement used to estimate some characteristics of the system that are difficult to measure or compute directly. The application of metrics in a blind simplistic manner is unlikely to yield any benefit. However, metrics can provide information to improve the actual quality of the system under development. *Metrics are guidelines, so it makes no sense to rigidly adhere to them. It makes more sense to use them to identify potential 'hot spots' or areas of potential concern* (Douglass, B, 2004).

Existing established metrics such as Nielsen's heuristics are not adequate enough to comprehensively analyse and judge the effectiveness of the design artefacts developed during this research. MetricMoIST is the resultant tool developed in this research to more accurately evaluate the artefacts. MetricsMoIST uses Checkland's 5 E's to evaluate the SSM component.

SSM's Five Es Performance Indicators for Decision Criteria

Since SSM is an integral part of the MoIST method, it is vital that its process integrity be evaluated. Checkland's SSM already has an inherent evaluation/check and balance system. This is known as the five (5) E's. These 5 E's are:

- efficacy (will it work at all?)
- efficiency (will it work with minimum resources?)
- effectiveness (does it contribute to the enterprise?)
- ethics (is it sound morally?)
- elegance (is it beautiful?)

The loose "process of engagement model" recognises the importance of related management and support activities. Root definitions, rich pictures and "idealised" solutions need to be in sufficient detail to enable practicalities and implementation issues to be evaluated. Resource demands and performance measures need to be articulated. However a complete specification is unlikely as the "ideal" is unlikely to be fully implemented - as it stands. Identifying implementation steps is the next phase.

[SSM's 5 E's]

SSM Product	Efficacy 20%	Efficiency 20%	Effectiveness 20%	Ethics 20%	Elegance 20%	Total 100%
Rich Picture						
Root Definition						
Conceptual Model						
Consensus Primary Task model -CPTM						

SSM Product	Criteria	YES	NO	Somewhat
RICH PICTURE	Rich picture describes the business situation under review			
	Presents a comprehensive view of the situation without being cluttered with too much detail			
	Relationships, including conflicts, between components are described			
	Roles of the people involved are indicated			
	Organisational issues are noted			
ROOT DEFINITION	The definition is well formulated and stands up to CATWOE analysis			
	The definition contains only one transformation			
CONCEPTUAL MODEL	Activities included in the model relate to words used in the root definition from which it is derived			
	The description of an activity begins with a verb			
	Monitor and control activities are included			
	There are activities to acquire and deploy resources			
	The activities and their logical dependencies form a coherent set			
CONSENSUS PRIMARY TASK MODEL	Activities included in the model relate to words used in the root definition of the test model			
	The description of an activity begins with a verb			
	The activities are described at the same level of detail			
	The activities and their logical dependencies form a coherent set			
	Activities to resolve conflicts have been included			

5.8 Limitations and challenges of the MoI ST me

thod & proposed solutions

One challenge lay in the fact that SSM models are fairly conventional without much room for deviation. On the other hand there is a plethora of UML models to choose from. Careful thought has to be made as to the selection of the method to be involved in the linkage (Dobbins, 1997). The key was to build SSM Conceptual models and generate information models from them in order to pass them to the UML Model. With the utilisation of the MoIST Method, a number of different models were generated. It was critical that consistency of each of these models was maintained. The nature of MoIST method dictates that each preceding phase acts as a foundation for the subsequent stage and feeds its output into it. This meant that any change in one model affected the other. These changes were usually easily seen and traced to ensure monitoring and control of the process. Consequently a change in the transformed models affected the UML model. This was one way of amalgamating the principles of hard and soft methods without epistemological damage to either (Doyle et al 1993). It is vital to seek to unite and amalgamate the best aspects of existing methods (Wood and Doyle 1989). MoIST is one such method that accomplished this successfully (see chapter 6).

5.9 Conclusion

Development of information systems in organizations will increasingly require solutions to problems of a wider nature than those traditionally addressed. This reflects the increasing penetration of information systems into the less routine areas of the organization. The breadth of knowledge of the systems designer is thus likely to grow. An emphasis will be on theory for guiding the particular type of development method to be used in a given situation. This is based on a more analytical approach to the organizational situation. ***Methods will have phases concerned with social and psychological factors in the problem situation,*** in conjunction with the more traditional phases characteristic of the hard approach (Flynn, 1998).

The MoIST method had to be applied in a real, live environment to determine its effectiveness. This empirical setting highlighted exactly how the MoIST worked under real conditions.

Chapter 6- Empirical Study

The problem is always to identify the problem. Too many people rush to solutions, and as a result they end up solving the wrong problem. How do you avoid that? By asking probing questions in an effort to expose the real issues; by challenging all of your assumptions and by collecting information even after you think you have identified the issue.

(John C Maxwell – Thinking for a Change, 2003)

6.1 Introduction

This description of my empirical work includes the area of application for this action research case study. The presented problem and the particular project domain studied are examined. Previous relevant experience is highlighted and shown to be of major benefit to this research. This chapter shows how the MoIST method was successfully used to implement a workable electronic system(ACcSys). Since the MoIST method is configured to be used most usually for development projects of an unstructured nature, SSM is used as the major 'finding out' mechanism in the preliminary investigation. SSM is also the model employed for the identification of issues, data collection in interviews and subsequent modelling and design are described. In the more specific data collection, the developed MoIST method is

progressively applied to the SSM findings. This is the design phase where the SSM's Conceptual Activities were linked to UML's Use Case models. Implementation of the system developed out of the MoIST's method application is also shown. A summary of the data in the light of the overall findings and implementation and user testing and evaluation are provided.

6.2. Area of Application

The Empirical Study concerned the School of Computing and Engineering at the University of Huddersfield. Huddersfield is an old textile mill town located in Yorkshire, in the North of England. Its population is approximately 500,000. Huddersfield itself nestles in the Pennine Hills and has a broad manufacturing and service base. The town is well known for its musical traditions and the world-famous Choral Society. The University of Huddersfield is a dynamic and expanding institution in a thriving West Yorkshire town. It has a friendly reputation, an excellent graduate employment record and offers a high level of student support. The University attracts students from all parts of the United Kingdom and over 60 countries world wide. There are currently over 17,000 students enrolled. Nearly 5,500 are full-time undergraduates. Over 3,500 are studying on full-time 'sandwich' courses with a year's work-placement in industry or commerce. There are almost 4,000 part-time undergraduates, 2,000 part-time postgraduates and over 800 full-time postgraduate students.

The University is organised into seven Schools: These are Applied Sciences, Education and Professional Development, Music and Humanities, Computing and Engineering, Huddersfield University Business School (HUBS), Design Technology, Human and Health Sciences. The School of Computing and Engineering is one of the seven Schools in the University. It has over 2000 full and part-time students on its own courses and teaches several thousand students in other Schools on service teaching contracts. The School has nearly 100 academic staff and in excess of 70 administrative and technical staff. There are three academic departments. Computing and Mathematical Sciences, Multimedia and Information Systems and Engineering and Technology.

In 2002, whilst conducting this research, there was a merger of the department of Computing with the department of Engineering. This amalgamation resulted in the School of Engineering and Computing. This merger had significant technological,

social and political implications. This change provided an even richer environment for the research findings to be tested and flourish.

Over the past twenty (20) years, the demographics of Huddersfield have changed to reflect a more multi-cultural persona. Huddersfield and Kirklees are the main feeder and 'catchment' areas for recruitment and admission at the University of Huddersfield. The new demographics influence the student expectations. For the university to better serve this student/client and to maintain its viability, the university teaching and learning practices have to reflect the changing characteristics of its students. These are some of the cultural and political factors that influenced this research.

6.3 Previous related action Research experience

Prior to conducting this research, I personally experienced the merging of two (2) university academic departments. Though this research was conducted in a different geographical setting, there was a general feeling of familiarity with the problem domain. This experience was as a lecturer in the School of Computing and Information Technology (SCIT) at the University of Technology, Jamaica. In 1997, the Department of Computing became the School of Computing and Information Technology (SCIT) and the Department of Engineering became the School of Engineering. These were then merged to become the Faculty of Engineering and Computing. This previous experience further cemented and solidified my competence base in the problem domain. I completed a Postgraduate Programme in Education (PG.Dip.). My Postgraduate research thesis addressed the problem of student aptitude for Computing subjects. Action research was carried out within the problem area. The major solution included the development of a new set of criteria for formal admission of Computing students to the School of Computing and IT. This research was requested by the School Administration who applied the recommendations. Subsequent research showed that the students were improving (Hopkins 1999). Doing an M.Sc. in Software Engineering and later M.Phil. level research at Bradford university also helped to hone and provide the confidence and competence needed to undertake PhD research at the University of Huddersfield.

Over the years of this research, informal and spontaneous conversations with faculty, administrative employees and students have proved very fruitful. These

conversations could be considered informal interviews for the purpose of the research. The diversity of people's philosophies and belief systems has influenced the successful outcomes of this research. It is always a privilege to be immersed in an organizational culture where ideas and changes incessantly flow and transpire all around and within to challenge and extend the boundaries of good thinking. The result in this research was beneficial change.

6.3.1 The General Problem Statement

The government has now mandated that fifty (50) % of the population must be given access to Higher education, (http://www.hefce.ac.uk). This means that persons who formerly might not have been considered worthy candidates for higher education will now be given the opportunity to pursue the same. This phenomenon has contributed to the problem of low retention that many universities and schools now face, (May and Bousted, 2004). Recent research has shown that low student retention and student withdrawal are primarily a result of unmet expectations and lack of student support. Discussions highlighted the importance of building peer support through academic and social activities. Data showed that students living in university accommodation had more peer contact and were significantly better retained than those who were not (May and Bousted, 2004).

Computing departments in the newer universities – post 1992 - developed a problem in the last ten years that has worsened. This is evidenced by the need for universities to employ staff to provide additional and in some cases remedial support for students. The problem is multi- faceted as there are issues of admission, retention and recruitment among others. The main problem is that students from non-traditional academic backgrounds tend to need help with a variety of study related issues in higher education (Sambell and Hubbard, 2004). There is a drive to examine not only ways in which students can be encouraged to continue their education to higher education level, but also that they are retained once there (May and Bousted, 2004).

Academic skills support policy is new in higher education as it was formerly perceived that there was no need for it. It was usually seen and expected in further education (FE). With the advent of the widening participation agenda, this support is

now vital for higher education (HE), (May and Bousted, 2004). Students at risk of non-completion or failure include mature students and students whose access is through non-traditional means. Compared to other undergraduates, these students have relatively weaker academic background skills and multiple problems, (Sanbell and Hubbard, 2004). These complexities have led to the employment of academic skills tutors in universities. The academic skills tutors primarily help students strengthen research, time management and study skills. They also provide diagnostic advice and support for specific learning disabilities. The majority of the current crop of students over the last ten years has entered university with poor study skills and many other learning difficulties. Some have exhibited low motivation and lack of basic study techniques and time management. Evidence of this is inherent or implied in the recruitment of additional staff, namely the academic skills tutors.

The University has recognised the need to deal with the pressing recruitment and retention issues. In order to address and correct the problem, academic skills tutors were appointed. The academic support project has been operational for one half of the stipulated timeframe. It was thought to be an opportune time to evaluate the support programme and its effectiveness. The challenge for the organisation is how to increase recruitment levels and maintain retention whilst simultaneously conforming to quality assurance standards.

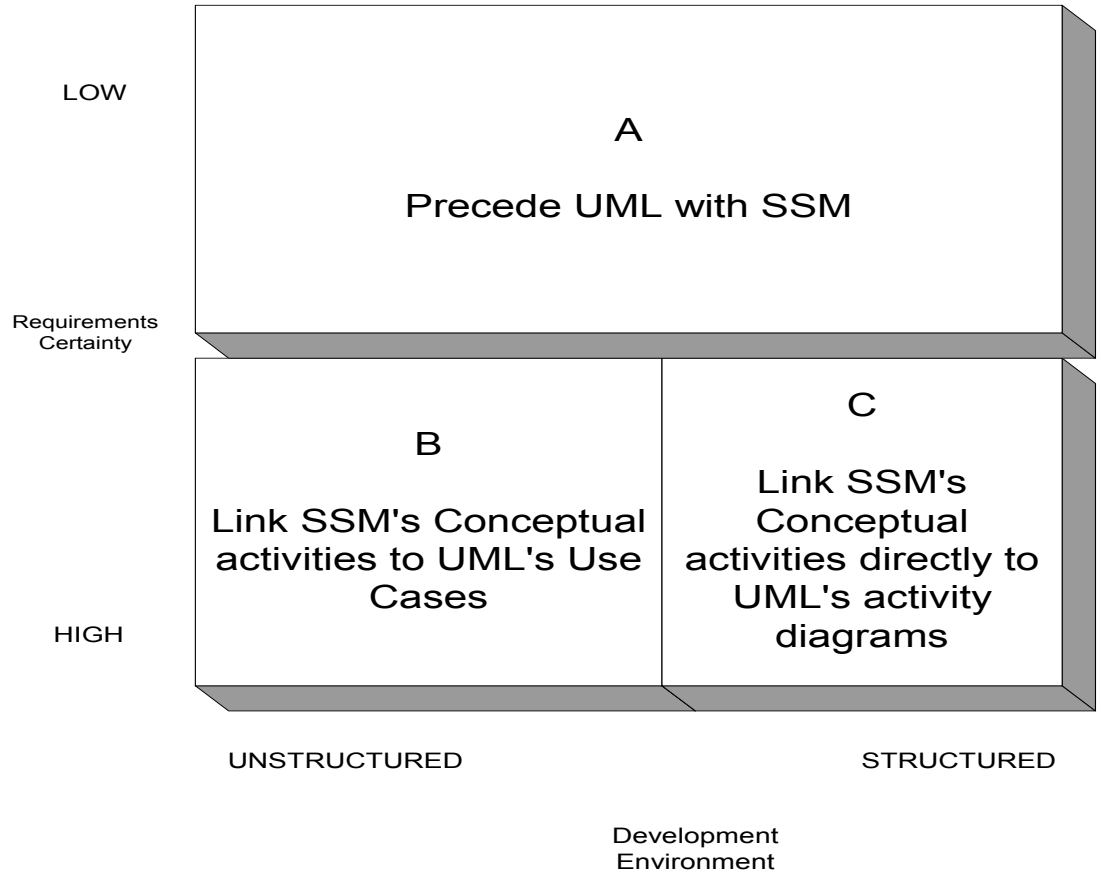
6.3.2 Description of how the current system works

The Academic Support tutor helps students with a number of study skills. These are namely literacy skills, referencing, examination techniques, report writing and time management. The Academic Skills Unit developed a paper based questionnaire (see appendix). This was used as a diagnostic tool for identifying students at risk of non-completion or failure. The questionnaire was disseminated to all new students at induction. The questionnaire responses were graded optically and the resultant grades categorised into three bands. Students whose grades fell in the lower band were contacted for further discussion and possible support. An Individual Learning Profile (ILP) was generated from the responses from each student and passed onto Pathway Leaders. This helped to identify students who were potentially at high risk of failure or non-completion. It also made the academic support process more of a school effort and not just the responsibility of the academic skills tutor. The first set of questionnaires was completed and the evaluation and administrative processes were carried out to ascertain the students who need extra-curricular support.

6.4 Empirical Study – Phase 1 – Analysis - Finding Out

Application of MoIST in the development of ACcSys

This phase represents the Analysis and 'finding out' phase of the research. It acts as the bridge between 'soft' Systems Science and 'hard' information systems design and implementation. The software development method used here is the method of integrating Systems thinking into Information Systems (MoIST). This MoIST Method combines the social and human factors of Soft Systems Methodology (SSM) with the technical framework method of the Unified Modelling Language (UML). It builds on the foundational work of (Davis, GB, 1982) and his Contingency Framework along with Multiview Method by (Avison and Wood-Harper,1990) and (CCTA,1993). The MoIST Method is designed to be specific to the combination of the SSM and UML methods. It is comprised of three (3) information systems development options. These are Options A, B and C. Option A precedes UML with SSM activities. Option B links SSM's Conceptual Activities to UML's Use Cases and Option C links SSM's Conceptual activities directly to UML's activity diagrams.



6.4.1 Using MoIST Project Option Selector Tool (MoPros) to select the best project option

The characteristics of the project were analysed using the MoIST Project Option Selector Tool in order to make the most informed decision. The points awarded are at the discretion of the project manager or could be through the development group consensus. It was found that the characteristics of the Academic Support Project were:

- Changes to business processes in the organization
- There is need for additional electronic solutions to alleviate the problem
- Organizational changes are likely
- The proposed system is to replace or enhance an existing system
- Development environment is unstructured
- Requirements are not relatively clear from the outset.

MoIST's Project Option Selection Tool (MoPros)

max. 25 points max. 25 points max. 25 points max. 25 points

Project Options	Types of users	Developers' skillsets	Organizational environment	General characteristics	Total
A	Users are a bit unsettled as they are experiencing organizational changes 20 points	Requirements at this point are not relatively clear to the development team 25 points	Development environment unstructured 25 points	Proposed system is to replace or enhance an existing system 25 points	95
B	Users uncertain about the need for the proposed system while others are more willing to be associated with it. 10 points	Requirements known at this point are relatively clear to 80% of the development team 0 points	Development environment has pockets of structured and unstructuredness. 15 points	Conflicting interests and the proposed system might cross functional borders 0 points	25
C	Users open to the new system	Requirements known at this information are very clear	Development environment is quite structured	Environment is relatively contention free	25

	0 points	to 90% of the development team 0 points	0 points	25 points	

Table 2: MoIST's Project Option Selection Tool (MoPros)

Using the MoIST Project Option Selector tool, it was found that the project requirements most closely matched project option A. **Option A** was chosen. Options B and C were not chosen at this point as they were not the best fit at this preliminary stage of the project. Given the project situation and its characteristics; the requirements were not specific enough to be directly linked to UML's Use Cases or UML's Activity diagrams.

MoIST Project Option A -Precede UML with SSM

Status: Requirements Certainty (Low) + Development Environment (Unstructured or Structured)

MoIST Option A's Activities
6. Requirements for computer-based information system 7. construct rich picture 8. develop relevant issue-based and primary task root definitions and conceptual models 9. derive consensus primary task model and information categories 10. formulate the recommendations for information system design

6.4.2 Mode of data gathering and interview data

The major client of the empirical study was the Academic Skills tutor of the School of Computing and Engineering. Interviews were arranged that afforded a variety of perspectives on the Academic Skills Support process.

After the interviews, SSM artefacts were constructed and subsequently showed to the client for feedback and amendments. At this point in the research, there was no idea for an electronic system. There was just the interest of learning about the situation and desiring to find ways to improve it from a research point of view. Initially learning in that area seemed to be exhausted and all the data was documented and stored away for future reference. That particular problem area was left and SSM studies were conducted in other areas of the school.

Several months later, the academic skills support tutors of the university were having one of their fortnightly meetings. They met to evaluate their effectiveness in their jobs. It was generally felt that they were doing well, but that an electronic intervention would help them increase effectiveness in identifying and supporting students at risk of non-completion of programmes. This led to invitations being extended to attend the next fortnightly meeting of the academic skills tutors. A formal presentation of the research was done of the perspective gained from doing an SSM analysis of the academic skills process in the school of Computing. Feedback was then provided by the other academic school tutors as to whether or not this perspective of their work was accurate. The presentation eventually became an interactive SSM session where some of the SSM artefacts were redefined as the dialogue expanded. The artefacts were subsequently amended to reflect this second, more comprehensive SSM view of the problem area.

6.4.3 Analysis using SSM

This describes the phase of the research where four (4) modelling activities were carried out. An overview of Soft Systems methodology was given. This formed the basis of the modelling carried out. It began with participatory soft systems modelling. This is where the clients were engaged in discussions to determine how the process worked. They provided the feedback necessary to the accuracy of the models. Non-participatory soft systems modelling was also done. This involved constructing the SSM artefacts and models after the sessions.

Soft Systems Methodology (SSM)

Soft Systems Methodology(SSM) was pioneered by Professor Peter Checkland (Checkland, 1981) at the University of Lancaster, UK. SSM seeks to represent unstructured situations with the primary goal being to understand the situation as it

really is. After understanding is gained, the methodologist or the owner is then empowered to make an intervention. This usually results in some improvement to the previous situation. Soft Systems Methodology (SSM) is used to learn more about the situation and to help to formulate solutions. SSM advocates cyclic learning of unstructured situations. SSM in its most basic form has several distinct stages. The first involves three different analyses of the existing situation. **Analysis one** deals with the intervention into the situation. **Analysis two** looks at the problem scenario as a social system. This is in terms of the behavioural norms and values that measure role performances as good or bad. **Analysis three** examines the politics and power distribution in the organization.

SSM Analysis Models

Analysis one gives a snapshot view of the major players in the study. It provides a succinct analysis of the situation being studied in the hope of making an effective and relevant intervention.

Analysis 1 – Analysis of the Intervention

Client	Academic Skills Tutor
Problem Solver	Soft Systems Methodologist
Problem Owners	Academic Skills Unit, University

Table 6.1: Analysis of the Intervention

Analysis two provides an overview of the social configuration of the problem domain. It delineates the roles of the key players. It also examines the behavioural norms of the role holders and highlights the values that determine satisfactory role performance of each player.

Analysis 2 – Social System Analysis

Roles	Behaviour norms	Values
Academic Skills Tutor	Helps students improve study skills	Delivery of academic learning support in the school - GOOD
Student	Learner enrolled at the university	Students not at accepted university standard - BAD
Pathway Leaders	Administrate university modules	Liaise with skills tutor - GOOD

Table 6.2: Analysis two

Analysis three examines and exposes any political undercurrents not explicitly defined in the problem domain. It summarises the analysis and details the disposition of power. That is who wields real power in the situation and therefore can make or break any system developed. It also examines the nature of the power described.

Analysis 3 – Political Systems Analysis

Disposition of Power

The Academic Skills tutor within the School of Computing had no formal supervisor or peer within the school. This provided reasonable autonomy to carry out support duties without necessarily having to wait for internal consensus.

Nature of Power

In the Academic Skills Unit, each school has its particular tutor. Among the tutors, nevertheless a hierarchy exists. Friendly professionalism seemed the existing atmosphere. Duties are allocated according to strengths and tutors seem to work together as a reasonably cohesive unit. The tutors meet every fortnight to discuss issues, strategies and problems. They work in conjunction with the disability office, library and the international office to achieve their aims. The possibility of power play in these external collaborations was not established.

After the analyses have been completed, a finding out or an investigation stage was then initiated. The artefact produced at this stage was the **Rich Picture**

SSM Rich Picture

The **Rich Picture** provides a snapshot of the entire situation as seen from the methodologist's perspective. It is a very effective modelling tool as it offers beneficial and ease of interaction with the client. It enables the methodologist to quickly get a more accurate view of the situation.

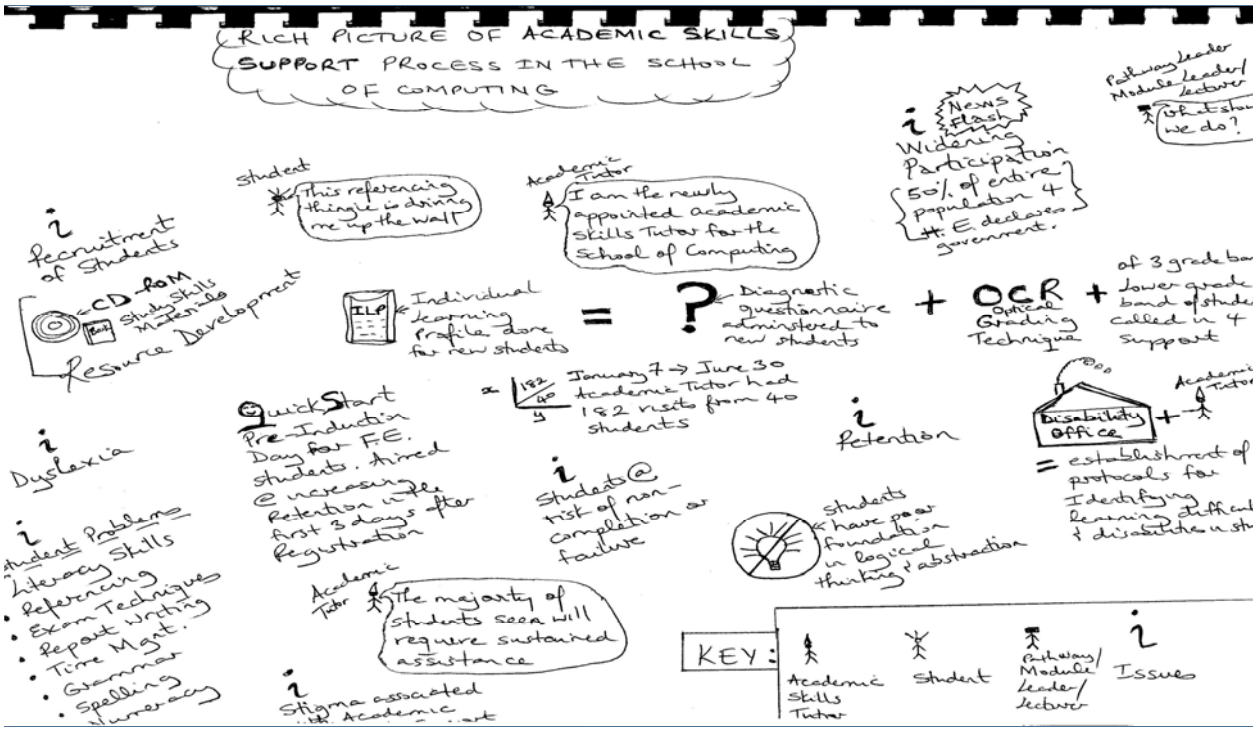


Fig 6.1: Rich Picture - Academic Skills Support (created by Hopkins, 2002)

The next stage involved constructing models of relevant human activity systems. These systems are then named and modelled and should not bear too much resemblance to the real world situation identified in the first stage. The **Root Definition** is subsequently formulated.

SSM Root Definition

The **Root Definition** is a brief textual statement that best describes the system and tells what the system will or should do. It acts as a sort of quasi-mission statement for the system.

A university learning and teaching innovation unit owned and professionally staffed system which, in the areas of academic skills technology, analyses and evaluates the academic competence of students, in order to identify students with problems at an early stage and to make proposals to the university learning innovation unit for re-

skilling and improving existing competence to enhance student passes within the resource limitations of the learning and teaching innovation unit.

Table 6.3: Root Definition of Academic Skills Support Process

CATWOE Model

C	Customer	Who would be victims/beneficiaries of the purposeful activity?	Whom
A	Actor	Who would do the activity?	Who
T	Transformation	What is the purposeful activity expressed as: Input–T–Output?	What
W	Weltanschauung	What view of the world makes this definition meaningful?	Assumption
O	Owner	Who could stop this activity?	Answerable
E	Environmental	What constraints does this system take as given?	Environment

Table 6.4: CATWOE elements of a root definition, Lai, 2000

CATWOE is a mnemonic used to ensure the 'well-formedness' of each **root definition**. CATWOE represents ***C***ustomer, ***A***ctor, *T*ransformation, ***W***eltanschauung(loosely translated as worldview), ***O***wner and ***E***nvironmental constraints. These represent the problem situation as perceived by the methodologist.

C	**Customer**	Student, Academic skills tutor
A	**Actor**	Academic skills tutor, lecturer
T	**Transformation**	Students in need of academic support → students better equipped academically
W	**Weltanschauung**	As many students as need it ought to have access to academic support to be able to successfully handle the given curriculum
O	**Owner**	Academic skills unit, University
E	**Environmental constraints**	Financial constraints for academic support programme 'at risk' students who do not want to participate in the support process

Table 6.5: CATWOE of academic support process

After the CATWOE had been applied to the Root definition, a **Conceptual Model** (CM) was derived.

SSM Conceptual Model

A **Conceptual Model** (CM) was derived from the root definition. This conceptual model provides activities that represent what was expressed in the root definition. A conceptual model could therefore be considered as an instantiation of a root definition. Here the conceptual model represents a transformation where the goal is to increase the number of successful students and reduce the number of students at risk of failure or non-completion.

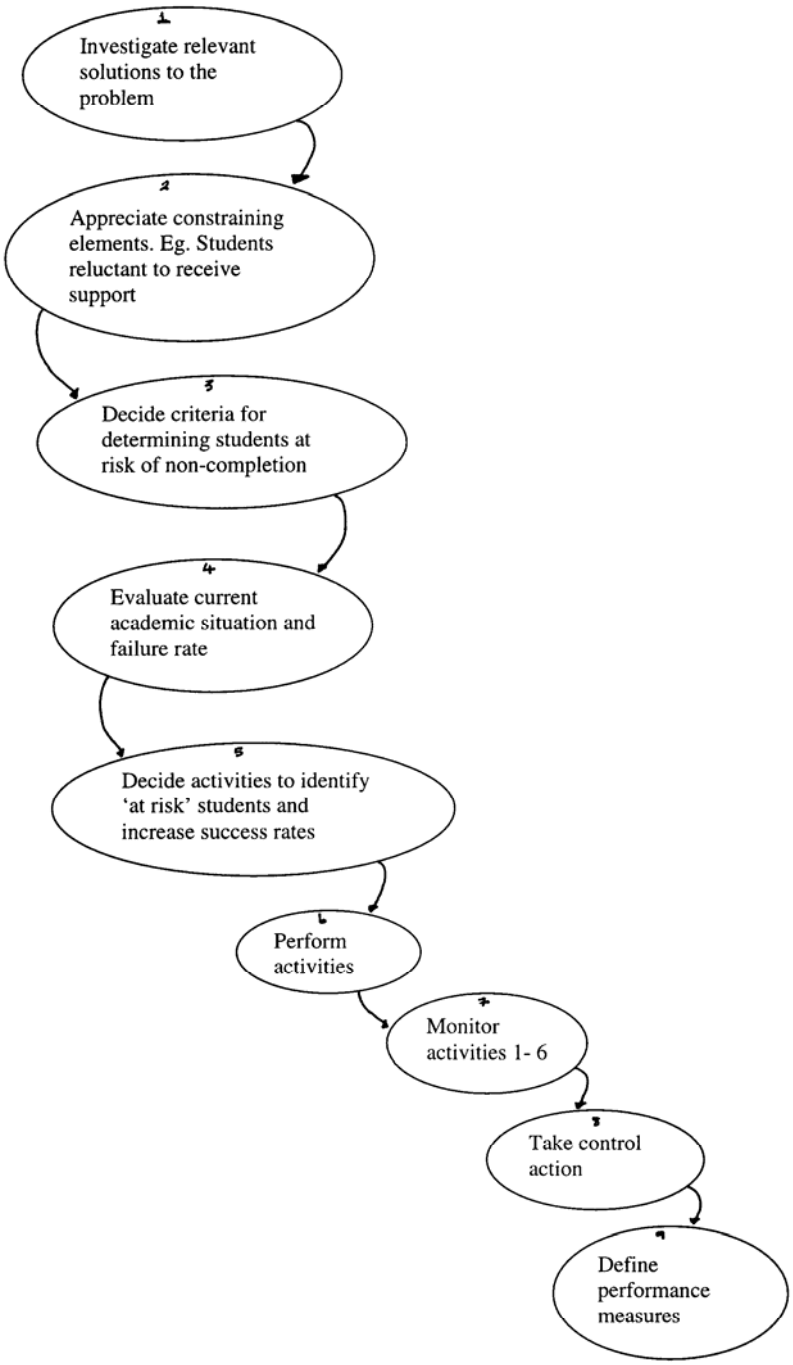


Figure 6.2: Conceptual Model of Academic Skills Support Process

The Conceptual model derived above offered a more high level view of the system. In order to get a lower level and more detailed picture, another conceptual model was constructed.

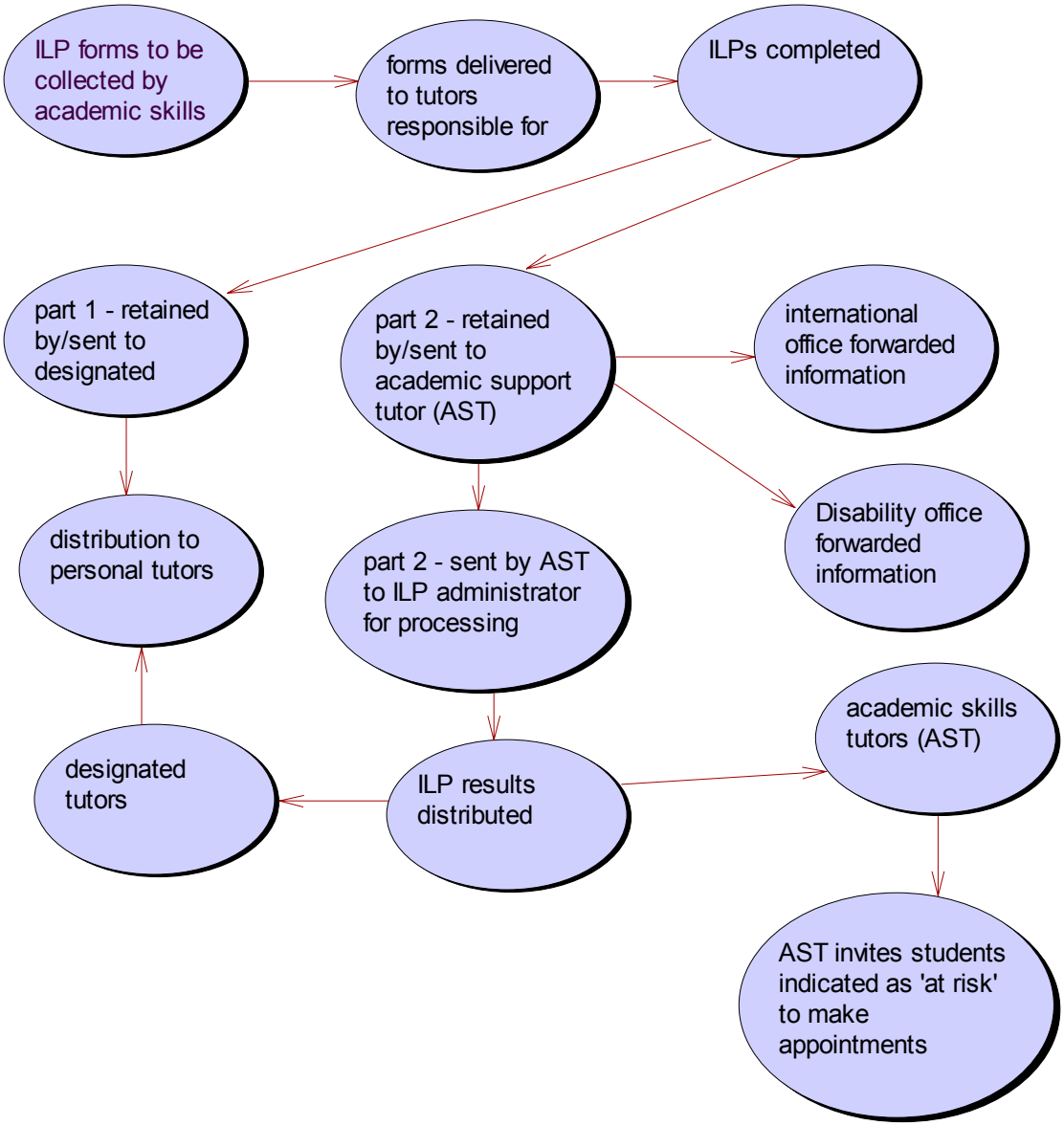


Figure 6.3: Specific Conceptual model of existing Academic Support Process

SSM Comparison Models

In this next stage, the rich picture derived in the first stage was then compared with the derived conceptual models and the tabular model below was constructed.

Conceptual	Reality
Investigate relevant solutions to the problem	This is being done to a reasonable extent
Appreciate constraining elements	Some constraining elements have been identified and are being addressed
Decide criteria for determining students at risk of non-completion	No formal criteria have been put forward by the academic skills unit
Evaluate current academic situation and failure rate	The evaluation process is ongoing
Decide activities to increase number of successful students	Some strategies to increase the success rate have already been implemented. For example the questionnaire as a diagnostic tool
Perform activities	The decided activities are currently being performed or are being fine tuned
Monitor activities 1-6	Monitoring is undertaken by the academic skills unit coordinators
Take control action	This can be improved
Define performance measures	Some have already been defined and as more becomes know, the definitions are extended

Table 6.6: Comparison Phase of Academic Skills Support Process

The findings were used to take action to improve the problem situation. The proposed changes are based on comparison between the conceptual model and the reality of the situation.

Defining Change

The proposed change is based on comparison between the conceptual model and the reality of the situation. It is thought to be systemically desirable and culturally feasible (Checkland, 1981). The solution to the highlighted problem lies in finding a

way to speed up the ILP administration process to ‘catch’ the ‘at risk’ students before they fall through the cracks.

An electronic means of linking student to pathway leader must of necessity be a part of the solution. An overall solution was therefore needed to make the process more efficient and thereby more effective. In true SSM style, this move was not wholeheartedly embraced by all the stakeholders as it was not feasible for every department. This was because most tutors thought that the majority of their new students would not be confident in using a computer at that stage and it might possible hinder the whole process instead of advancing it. Subsequently only the Schools of Computing and Engineering and Business expressed interest in this online administration of the ILP questionnaire.

6.5 Empirical Study Phase 2 – from Analysis to Design

This phase represents the design phase of the electronic system. Here the SSM outcomes are mapped to UML. The MoIST Project Option Selector tool is again used to evaluate the project and select the next most suitable project option.

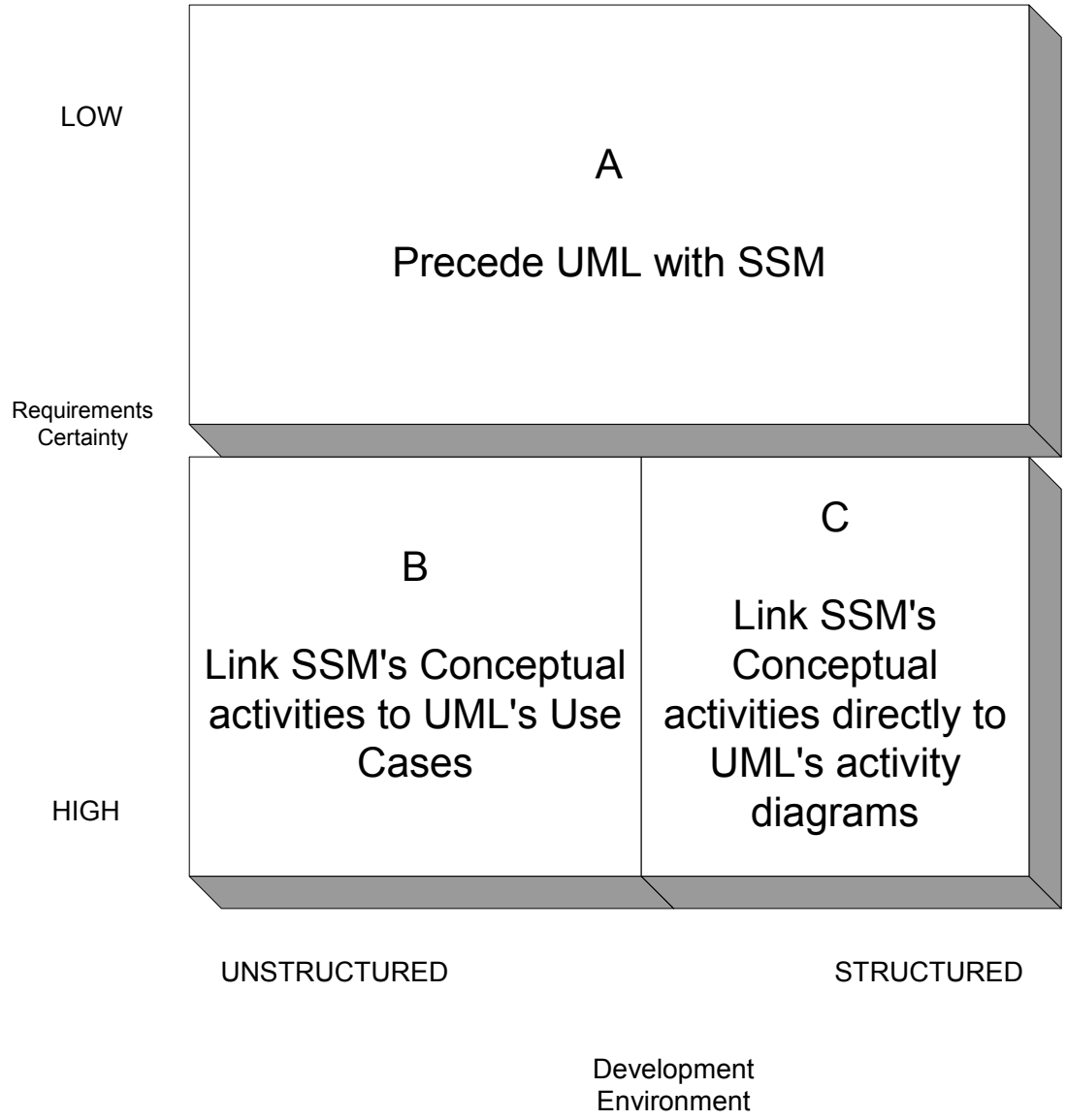


Figure 6.4: The MoIST Method

6.5.1 Using MoIst's Project Option Selector Tool to get the best project option

After the SSM findings, the characteristics of the project were again analysed using the MoPros Selector Tool in order to make the most informed decision for this stage of the development process. The points are awarded at the discretion of the project manager or by general consensus with the development team. It was found that the characteristics of the development project thus far were:

- Users uncertain about the need for the proposed system
- Development environment has some existing pockets of structure and unstructuredness

MoIST's Project Option Selection Tool (MoPros)

max. 25 points max. 25 points max. 25 points max. 25 points

Project Options	Types of users	Developers' skillsets	Organizational environment	General characteristics	Total
A	Users are a bit unsettled as they are experiencing organizational changes 0 points	Requirements at this point are not relatively clear to the development team 0 points	Development environment unstructured 10 points	Proposed system is to replace or enhance an existing system 10 points	20
B	Users uncertain about the need for the proposed system while others are more willing to be associated with it 20 points	Requirements known at this point are relatively clear to 80% of the development team. 22 points	Development environment has pockets of structure and unstructuredness. 19 points	Conflicting interests and the proposed system might cross functional borders 0 points	61
C	Users open to the new system 10 points	Requirements known at this information are very clear to 90% of the development team 0 points	Development environment is quite structured 10 points	Environment is relatively contention free 18 points	38

Table 6.7: MoIST's Project Option Selection Tool (MoPros)

Using the MoIST Project Option Selector Tool, it was found that the project requirements most closely matched project option B. **Option B** was chosen since Option A's activities were already completed within the SSM study conducted. Option C was not used as the developer thought that given the project situation and its characteristics; the requirements were not specific enough to be directly linked to UML's activity diagrams.

Option B: Enhance inception stage with SSM by deriving use cases from activities within the conceptual model.

Status: Requirements Certainty (High) + Development Environment (Unstructured)

MoIST Option B's Activities	
B1.	**Derive conceptual primary task model (CPTM).**
B2.	**Select and prioritise Conceptual model activities.**
B3.	**Determine which activities require further decomposition.**
B4.	**Determine which of the selected activities are candidates for IT support.**
B5.	**Identify actors.**
B6.	**Develop high-level use cases.**
B7.	**Develop multi-level use cases.**
B8.	**Identify high-level objects.**
B9.	**Map required high level services onto objects.**
B10.	**Continue design.**

Fig 6.4.1 Application of Option B within the MoIST method

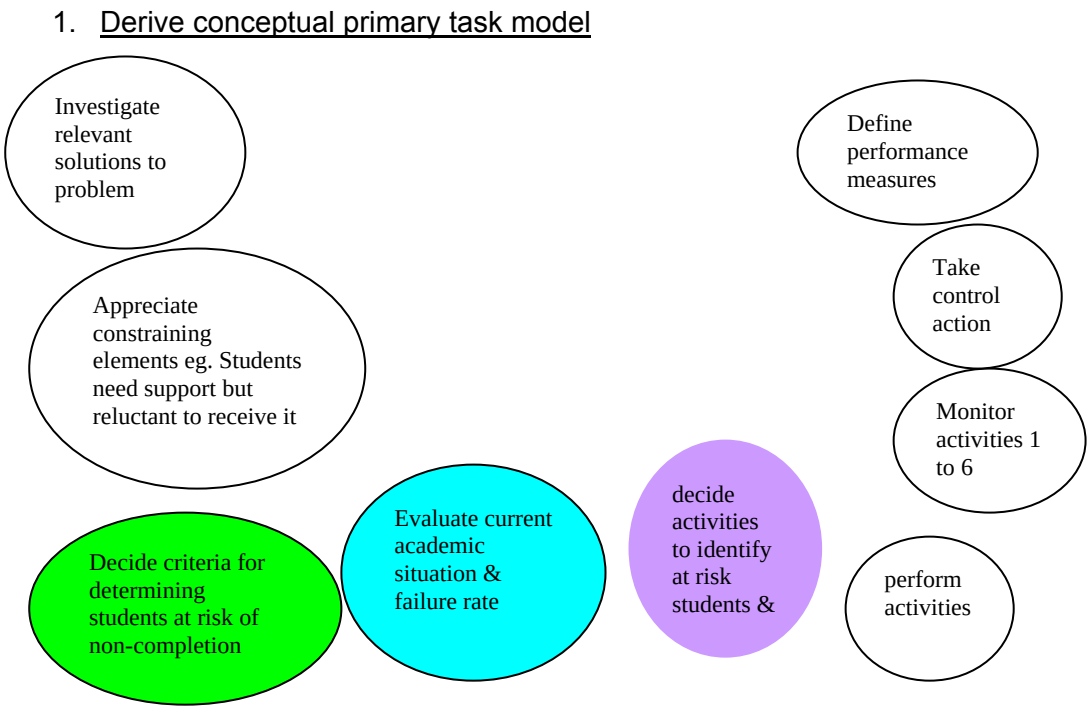


Figure 6.4.2: Conceptual model was derived from the SSM finding out stage.

2. Select and prioritise Conceptual model activities to be investigated for possible IT support

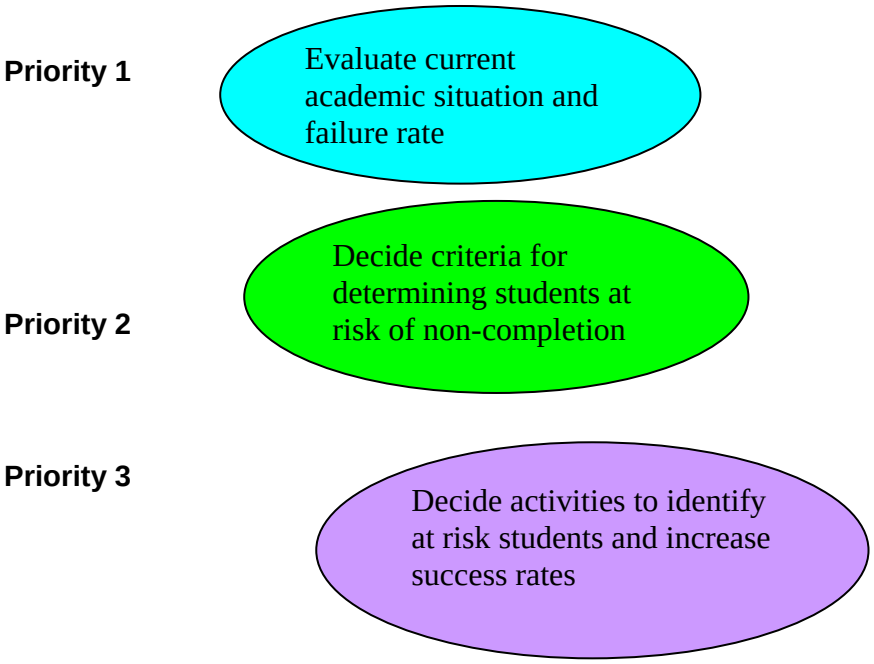


Figure 6.4.3: Priorities 1,2 and 3

3. Identify scope or scale of OOA by determining which of the selected low-level activities are likely candidates for IT support. Also determine which may require further decomposition of some specific activities for which responsible use of IT is unclear

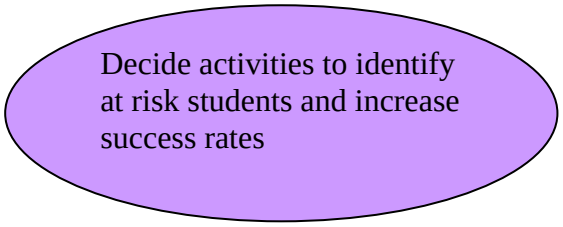


Figure 6.4.5: Activity in Option B of the MoIST Method

The selected low-level activity above is the most likely candidate activity for IT support

4. Identify actors for each of the low-level activities.

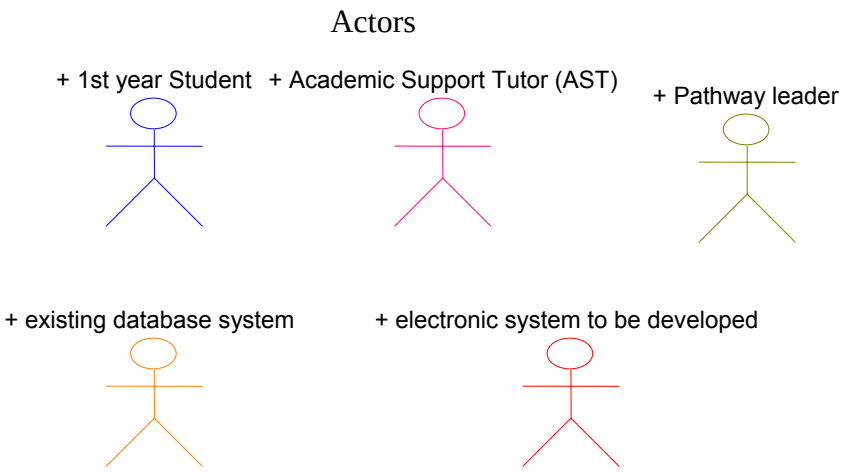


Figure 6.4.6: Actors for Low-Level Activities

5. Develop top level use cases. Let each identified low level activity serve as the name of a use case. Involve the relevant actors and /or domain experts when writing up these top level use cases.

UML Use Case Model

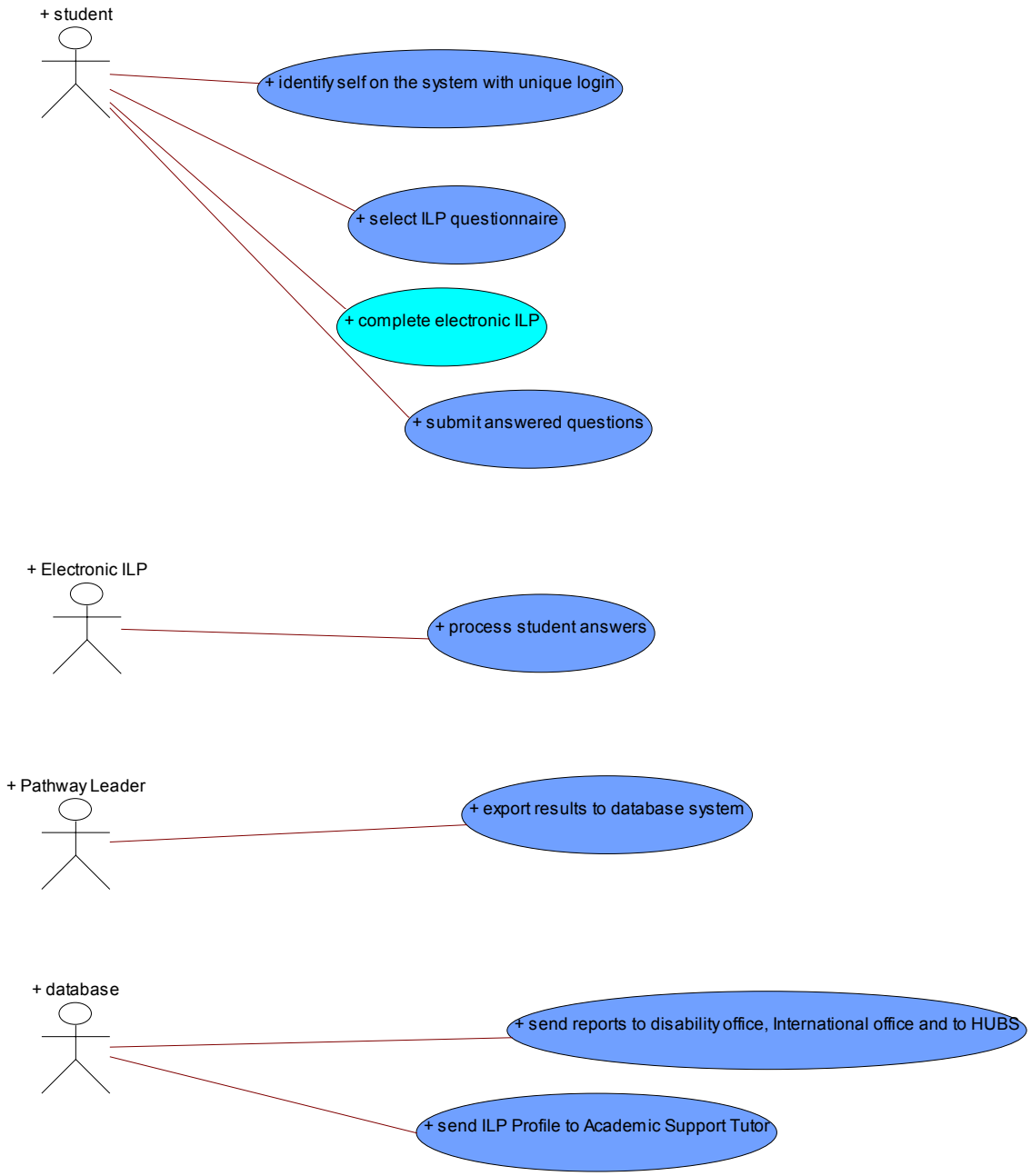


Figure 6.4.7: UML Use Case Model

6. Identify high level objects. Identify the objects from the use cases and develop class diagrams and association are made between the objects to express their relationships.

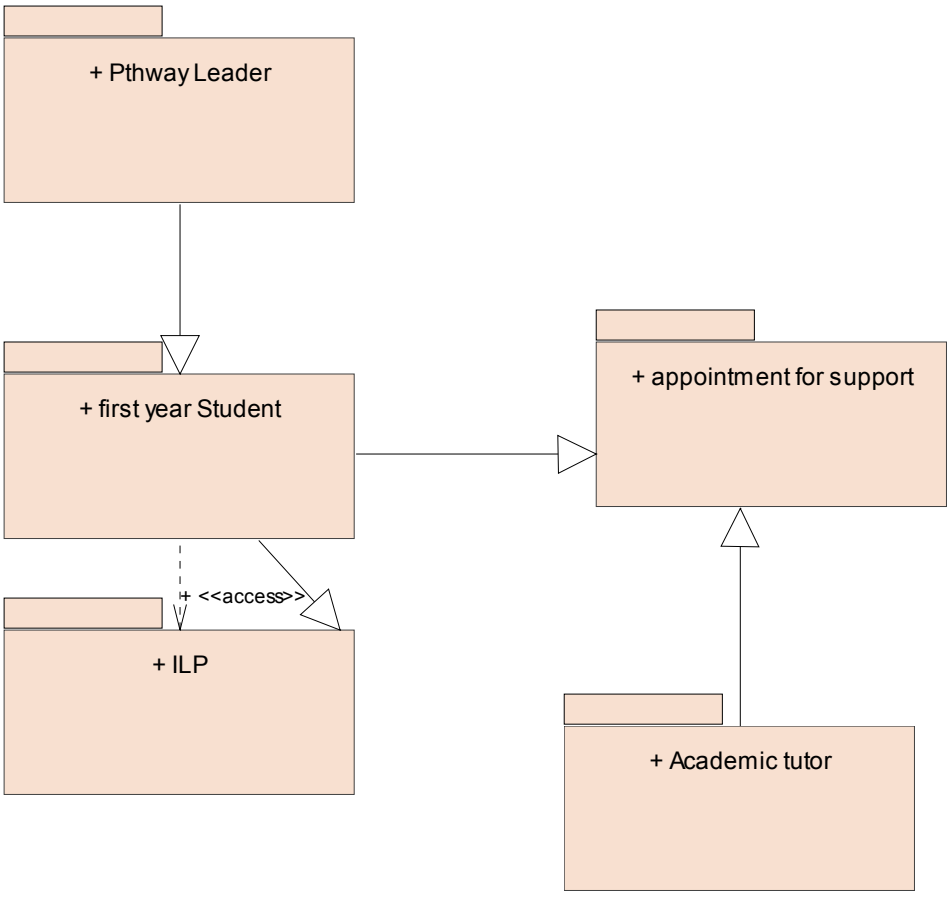


Figure 6.4.8: Class Diagram

CASE Tool used in Design phase

QSEE Superlite was the CASE tool used in this research. It is a generic modelling environment that supports a large number of applications. It was designed by QSEE Technologies Ltd – 2001-2004. QSEE multi-CASE is a collection of sub-tools designed to aid in the analysis and design of software systems. The tool allows a user to combine over a dozen analysis and design approaches to help identify and solve software related problems. Some of the models created using the software include Rich Pictures, Conceptual models, flowcharts, UML models, state transition diagrams, data flow diagrams, entity relationship diagrams and structure charts.

Fig 6.5: screenshot of CASE tool

This tool was the most ideal one for this research as it facilitated the creation of relevant models from both the Soft systems and hard systems paradigms pertinent to the research.

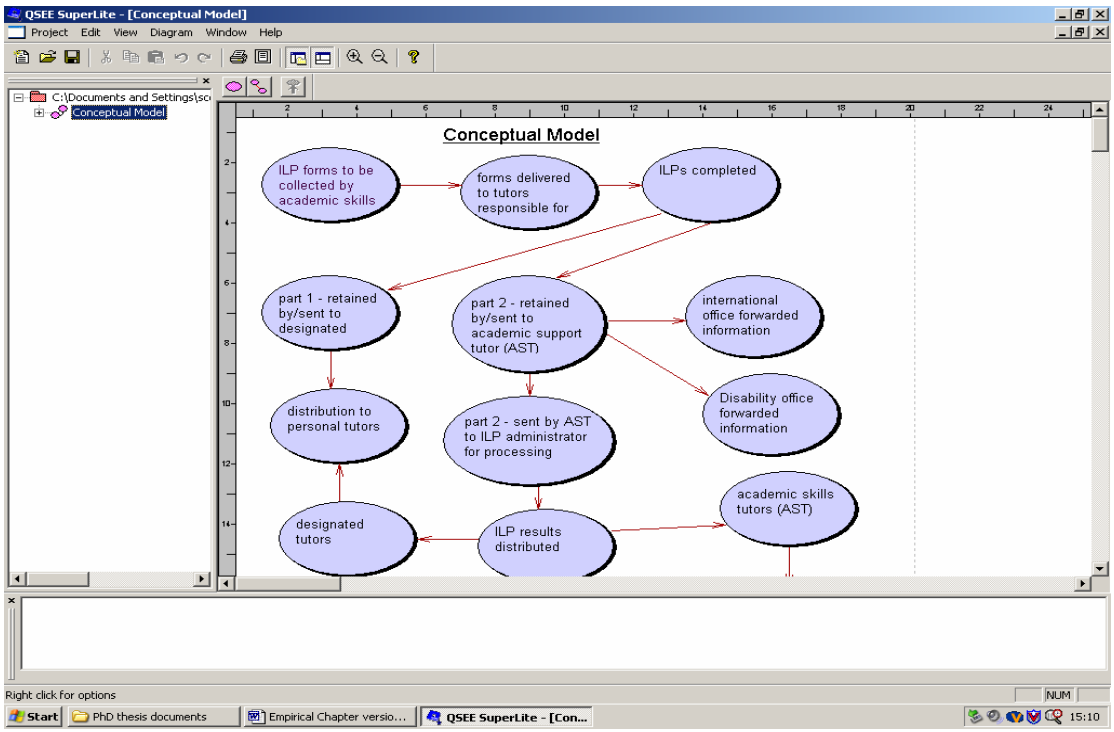


Fig 6.5.1:screenshot of conceptual model created during the research using QSEE Superlite

After Option B of the MoIST method was applied to the SSM results. This marked the end of the design phase. The design artefacts were then evaluated below using the Metrics MoIST evaluator system developed during the research.

6.6: Empirical Study Phase 3- from Design to Evaluation (using MetricsMoIST)

MetricsMoIST is a heuristic tool developed as part of this research. It uses Checkland's 5 E's to evaluate the SSM component.

(1) Checkland's 5 E's

These were used to evaluate the integrity of the results of the SSM study conducted in the first part of the empirical research. Each 'E' was scored on a scale of one (1) to ten (10).

- efficacy (will it work at all?)
- efficiency (will it work with minimum resources?)
- effectiveness (does it contribute to the enterprise?)
- ethics (is it sound morally?)
- elegance (is it beautiful?)

(a) ***Efficacy***. This measured whether or not the ACsSys worked at all. It gained full measure here as it was used in a real, live situation and it worked well and produced the desired results – 9/10

(b) ***Efficiency***. This measured whether ACcSys would work with minimum resources. It worked with Question Tools software loaded onto shared network resources. The development project did not have very many team members and it produced a workable electronic system. - 9.5/10

(c) ***Effectiveness:*** This measured the level of ACcSys's contribution to the School of Computing and Engineering. It made quite a substantial contribution as it produced turnaround in one (1) day what took several months using the previous paper based system. – 8/10

(d) ***Ethics:*** This measured the moral soundness of ACcSys. Ethics may be defined as acting fairly and in accordance with existing regulations and policies. ACcSys did not violate or break any established ethical code. - 8/10

(e) ***Elegance:*** this examined the measure of beauty in ACcSys. This is a very subjective criterion. There is room for improvement in this aspect as the software had predefined templates that did not allow extra room for creativity and innovativeness in its aesthetic design. -5/10

Overall evaluation of initial Performance Criteria and System characteristics

Performance Criteria	Evaluation of successful performance of initial criteria
Electronic implementation of paper based version of Individual Learning Profile (ILP)	The paper based version of the ILP was successfully converted to an electronic format that was user friendly and retained the scores for purposes of further analysis and data manipulation.
Implementation to be done before or during the first 2 weeks of academic term	The implementation was completed in time for its live run during induction week. The fast implementation time was due to the user friendliness of the software used
ILP scoring outcomes processed electronically	The software used to implement the ILP electronically had an inbuilt mechanism to retain and process the scores electronically
Three lists of students identified: 1. those declaring a learning difficulty/disability. 2. Those home and students from EEC countries requiring additional English language support. 3. Those students not home or EEC requiring additional English Language support.	This identification was carried out by the back-end system already in existence. The results were sent in comma separated format (CSV) to the back-end system. The data was then manipulated to generate the lists.

System characteristics	Comments
Timeframe	A complete system was expected to be up and running by the 3rd week in September 2004
Staff	the academic skills tutor and existing lecturers have been trained to use the system
Budget	Development costs were not expected to exceed

The MoIST method is a 'design' method. Its purpose is to reinforce the breadth and depth of the analysis to ensure that the right design goals are met. This research however was so successful that after the MoIST method was comprehensively applied and evaluated, it was found that there was a case for provision and implementation of an electronic solution. This solution ACcSys was the front end for an existing electronic system devised by a staff member in another school. After the development of ACcsys, work was done to achieve a seamless interface between the two systems. This was successfully done. The steps involved in implementation are detailed below.

6.7: Empirical Study Phase 4 – from Evaluation to Implementation

6.7.1 Exploration of several implementation solutions

Meetings were held with the university academic skills tutors and with the Head of tutors to discuss the desired electronic intervention. One initial idea was for a tracking system each academic skills tutor would be able to access at any given point in time. This would enable the tutor to see if a student had 'dropped out' of the academic system or if they had been retained. Work was started on this. Later yet another idea emerged for a Web Tracking System. This was to be modelled on the existing on-line School of Computing and Engineering web interface. The early plan was that this would draw on existing student data from the current Applicant & Student Information System (ASIS) used by the entire university.

While this was being explored, other meetings were held with the Computing Academic Skills Tutor. These meetings highlighted existing problems within the current academic skills process. The Academic Skills stakeholder requested specifications for a system that would enable academic skills tutors' to execute their professional duties more efficiently. The core of these duties was essentially to diagnose, identify and provide support for students at risk of non-completion of

academic programmes. This request changed the dynamics of the system being worked on somewhat. The client later decided that the need to track at risk students was secondary to the need for an electronic diagnostic tool where the outcomes could be scored automatically. That was somewhat disappointing as work had already been started on a prototype of the tracking system. That was however the nature of systems analysis and design. Work was nevertheless started on a system that would achieve what the client wanted. This meant that the existing paper based ILP questionnaire would be transformed into an electronic format. There was another meeting of the university skills support tutors. Most tutors with the exception of two (2), decided that an electronic ILP would not be suitable for their academic schools. One unanimous reason cited was that students might be turned off from the technology and this would defeat the essential purpose of the ILP. This purpose is to determine students' level of academic competence using specified determinants.
Schools of Computing and Huddersfield Business school were the only two (2) that expressed an interest. A formal commissioning was then given by the clients, the computing academic skills tutors.

6.7.2 Commissioning of the electronic system by the client

This commissioning was given to develop the agreed electronic system by the client. Further to previous discussions and conversations, a project management meeting was held with all the relevant personnel in the academic support process. From that project meeting, it was decided that the administration of the ILP would not proceed electronically as a collective entity. If individual schools wished to proceed with piloting a prototype, then they were free to do so. Consequently School of Computing and Engineering and Huddersfield University Business School expressed their interest in the development of a prototype electronic version of the ILP in their respective Schools. This was to be piloted from September 2004. Those schools which decided on continuing the paper based administration of the ILP questionnaire had a meeting in which the form was updated and upgraded to reflect a more modern format.
The electronic version was required to meet the following specifications as detailed by the client:

- On-line completion of the form with a paper-based version also available. The point at which students would complete, and in what manner, would have to be determined, but would need to take place before or during the first two

weeks of term, at the latest. On the paper-based version this is currently undertaken at pathway induction meetings

- Scoring outcomes processed electronically

- The student details/scoring outcomes automatically linked to those of both pathway leader and first year tutor

- Students declaring a learning difficulty/disability, those requiring additional English language support and scoring outcomes with two or more sections less than 15 identified, together with details of individual students, passed to both pathway leader and first year tutor

- Three lists of students identified: 1. Those declaring a learning difficulty/disability. 2. Those home and students from EEC countries requiring additional English Language Support. 3. Those students not Home or EEC requiring additional English Language support.

At the end of the research the above specifications describe the features of the new system to be developed and forms the criteria by which the new system will be judged to be successful.

6.7.3 Further exploration of several implementation solutions

The thinking, experimenting and exploring the best way to achieve the desired electronic system began again in earnest. At first, Blackboard version 6 seemed to be the tool of choice to facilitate the clients' specification. It was eventually decided that Blackboard version 6 did not have the pertinent characteristics needed to produce the expected system. Question Tools software was then considered as a more viable option. Question Tools (QT) is a new computerised assessment tool which was installed in the School of Computing and Engineering during 2003-2004. In depth analysis of Question Tools revealed that it had a greater percentage of the functionality needed to provide an electronic version of the ILP. It possessed the capability to produce data in a form acceptable as input to the analysis database. It also facilitated the provision of reports in a parallel format as the ones in the last academic year. The electronic ILP was subsequently developed using Question Tools software. Bottlenecks in the process were dealt with. The electronic ILP was

now ready for induction of new students. It was way ahead of the finish date required by the client.

ACcSys – Introduction

ACcSys is an electronic system that facilitates the retrieval, storage and analysis of student data. This data is analysed by the system and sorts the students into designated categories. This information is then used as a diagnostic tool to identify students who are 'at risk' of not completing their registered programme. The ACcSys helps to quickly identify those students who need help in specific areas and allows that help to be provided on time. Consequently this also helps to improve student retention.

ACcSys platforms

ACcSys will run on a wide variety of hardware platforms. A typical system would be an Intel processor based server running a mixture of MS thin clients and Nec terminals. This is preferred as it offers a very stable platform that is reliable. Uptimes greater than 300 days are the norm. It is also efficient, very scaleable and portable and most importantly is not susceptible to viral attack owing to the VShield software set up.

Web Enabled

ACcSys works very well with simple web browsers. As well as working with Explorer and Netscape, it can also be used with the new generation of embedded web browsers. This allows users access on the move with a mobile phone and a PDA.

Specifications sample of actual platform used for successful live ACcSys run

Make	Proc	Speed	Memory	HD(Gb)	Graphics	Sound	CD/DVD
NEC	P4	2.8	512	80	GeForce FX 52000 AGP 128Mb DDR	On board	RW combo
Stone	P4	2.2	512	80	On board	Sound blaster live 5.1	RW combo

Table 6.8

Design of ACcSys electronic system using Question Tools

Question Tools (QT) is an integrated suite of products that facilitates the creation of online lessons, exercises, surveys, tests and exams. It automatically collects and analyses results.

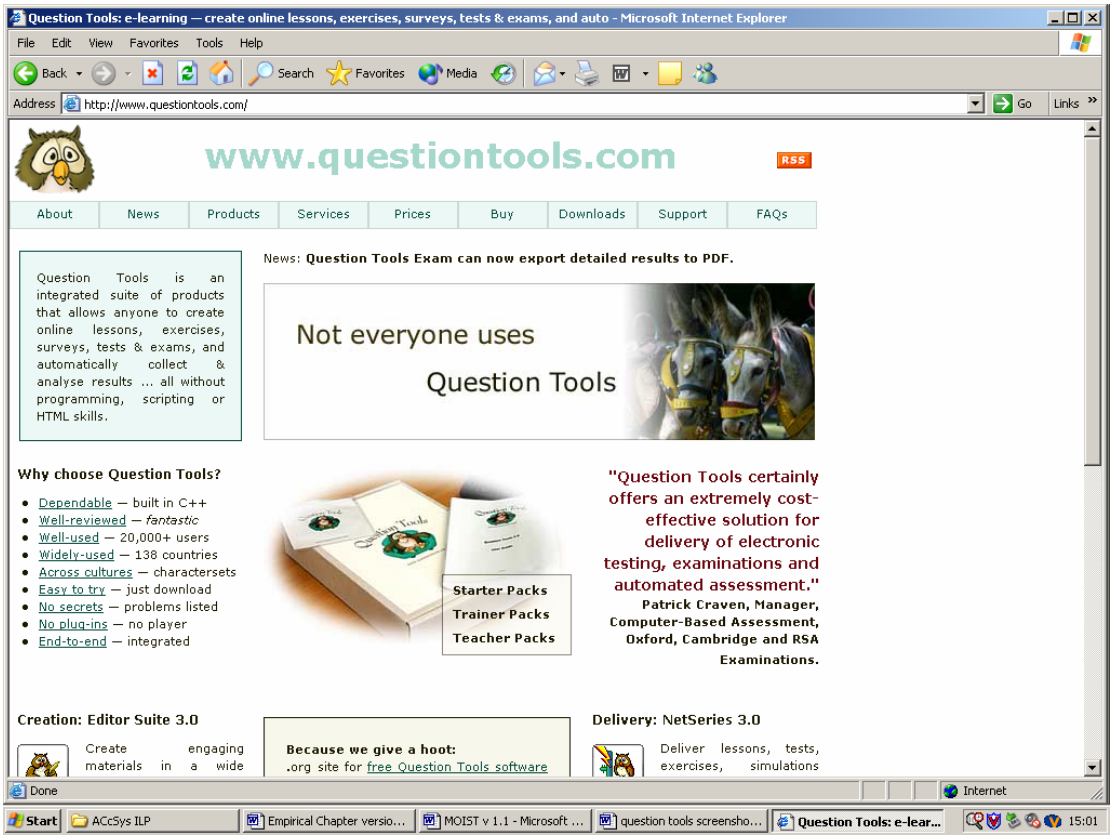


Fig 6.14 Screenshot of Question Tools Website

The Question Tools Editor was used to derive the ACcSys. The ACcSys Electronic Individual Learning Profile (ILP) is used to get all students details in one central repository. This makes it easier to manipulate, access, sort and report on the common data. It is therefore excellent for ease of use and access. The ACcSys has 59 screens in total. Part one allows for entry of student details. Part 2 of the ILP itself is composed of 8 sections. These are ‘Speaking and Listening’, ‘Reading and Researching’, ‘Writing’, ‘Language’, ‘Disability/Special need’, ‘Time Management’, ‘Numeracy skills’ and ‘Information Technology skills’(see appendix for all the ACcSys screens).

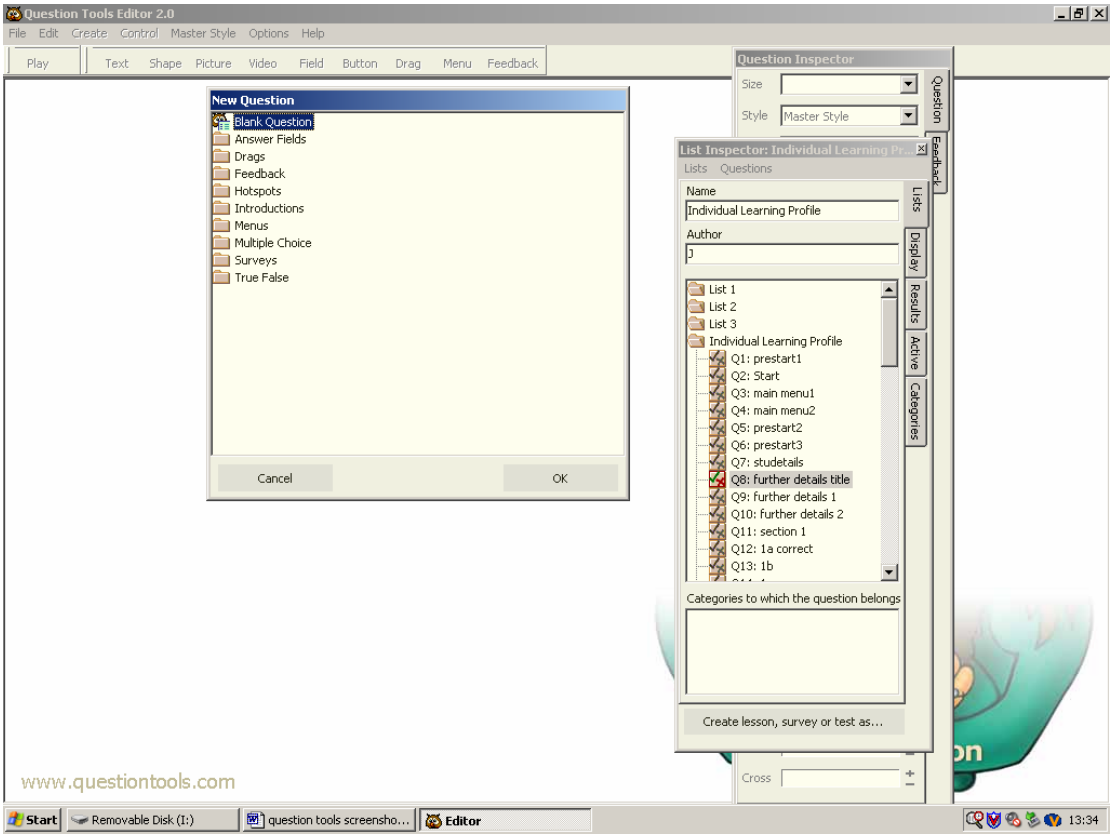


Fig 6.15 This is the QT editor with New Question option dialog box

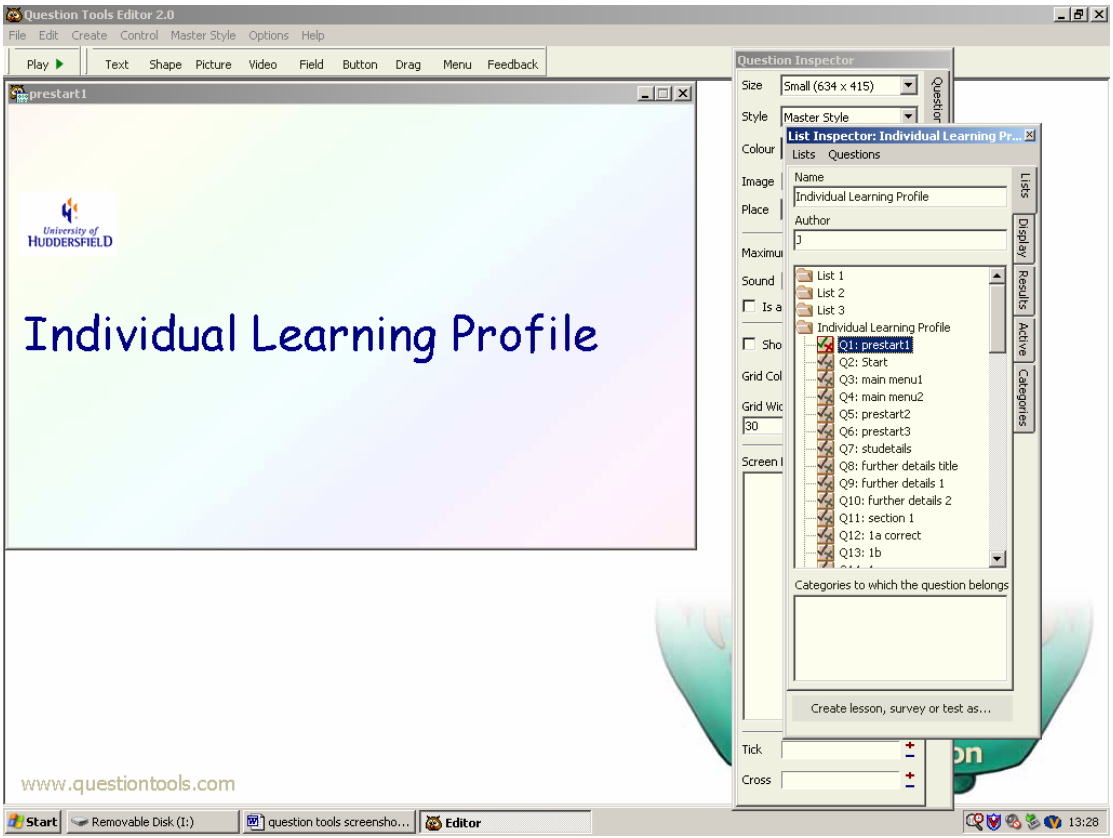


Fig 6.16 This is the first screen that the student sees with the title ‘Individual Learning Profile’

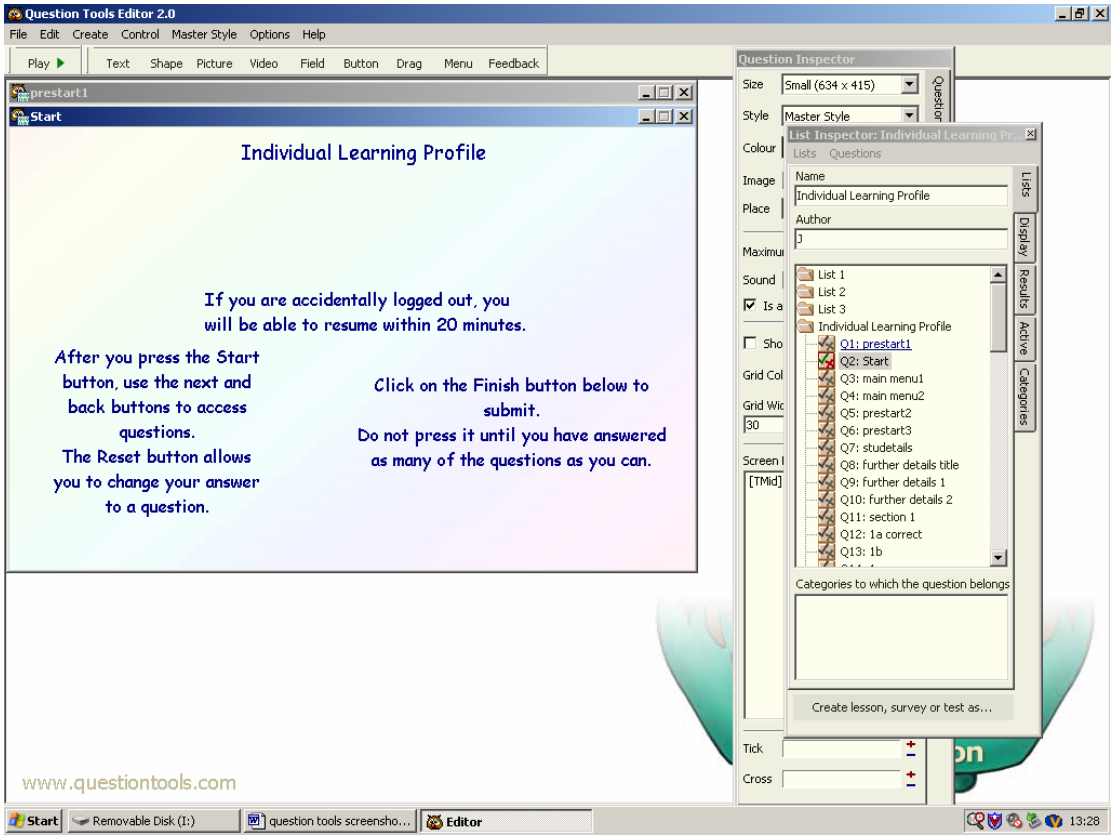


Fig 6.17 Instructions screen for user navigation through the electronic ILP profile

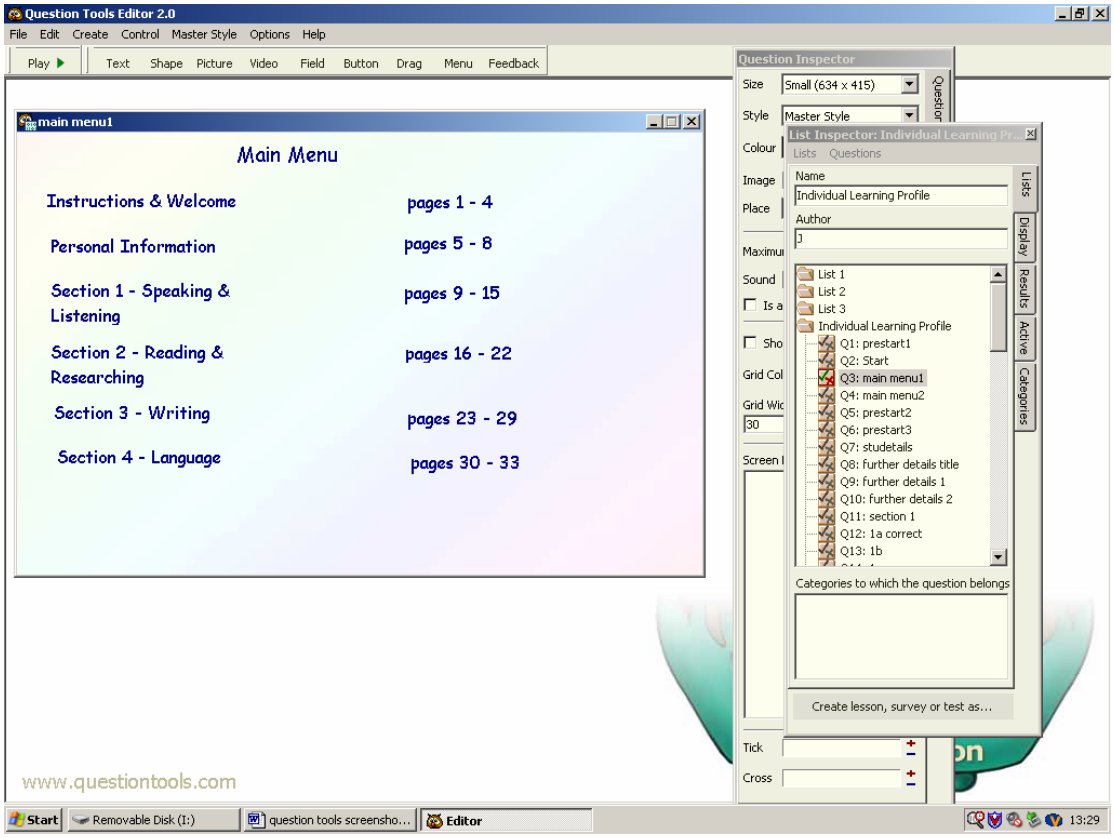


Fig 6.18 Main menu makes the system more user friendly and organised for the user to follow

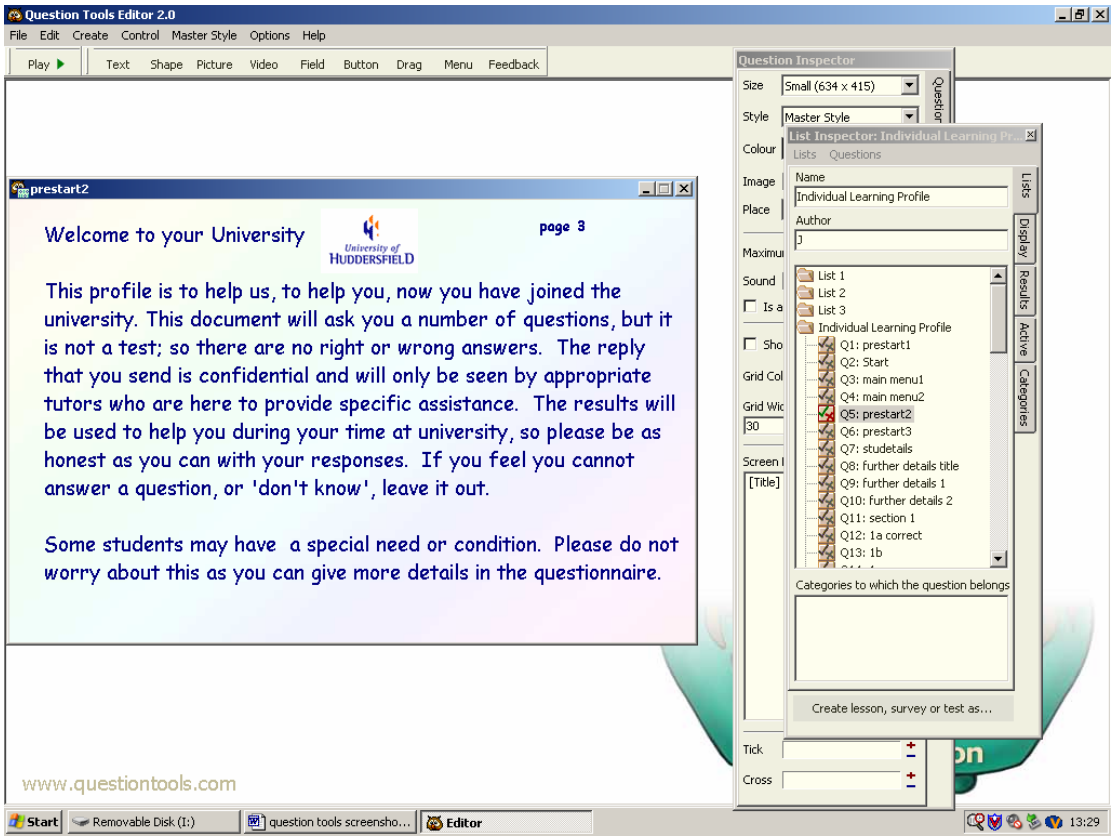


Fig 6.19 Welcome screen explains the purpose of the system and attempts to relax the user

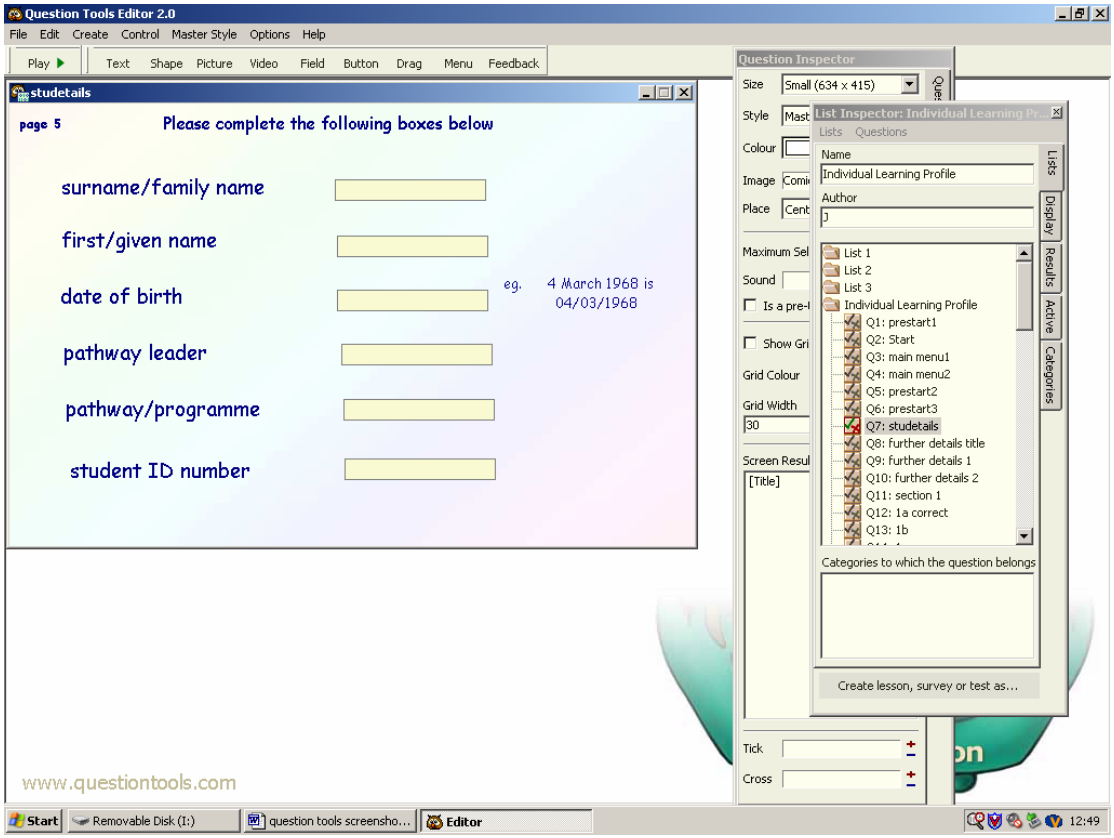


Fig 6.20 Student details screen allows for entry of relevant student data

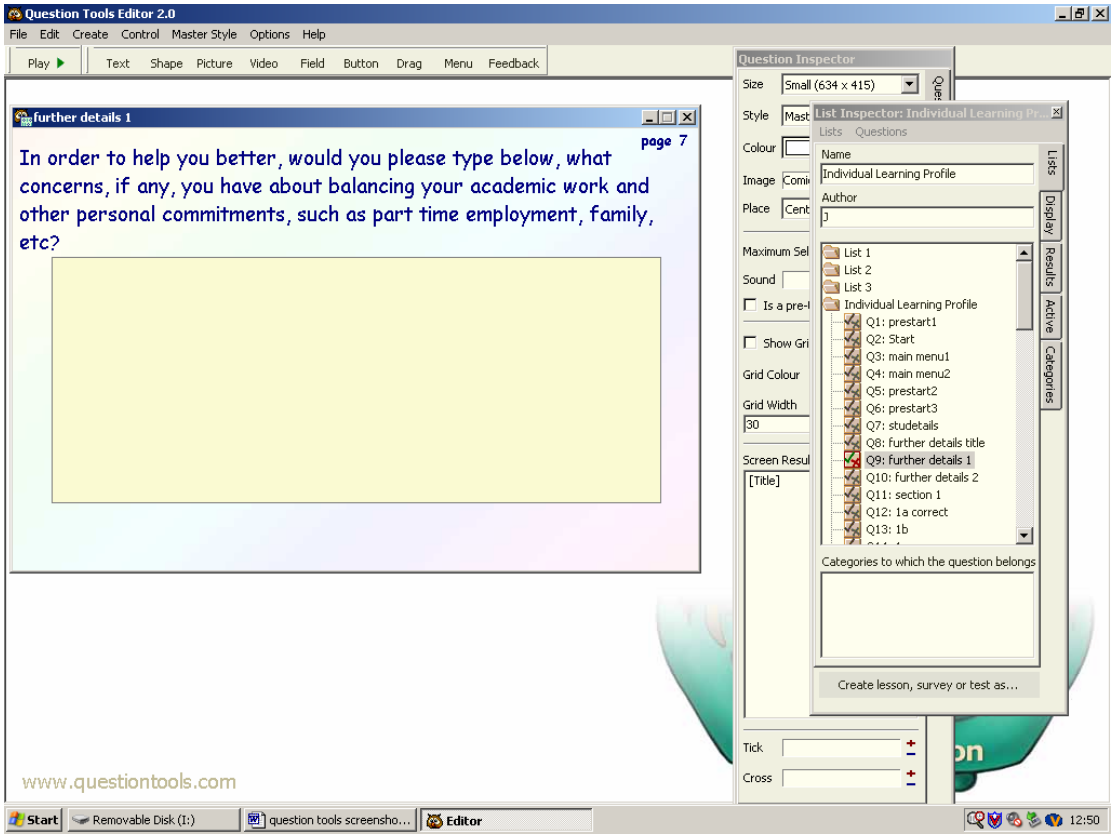


Fig 6.21 Further details screen tests their writing abilities as it is free form writing in sentences

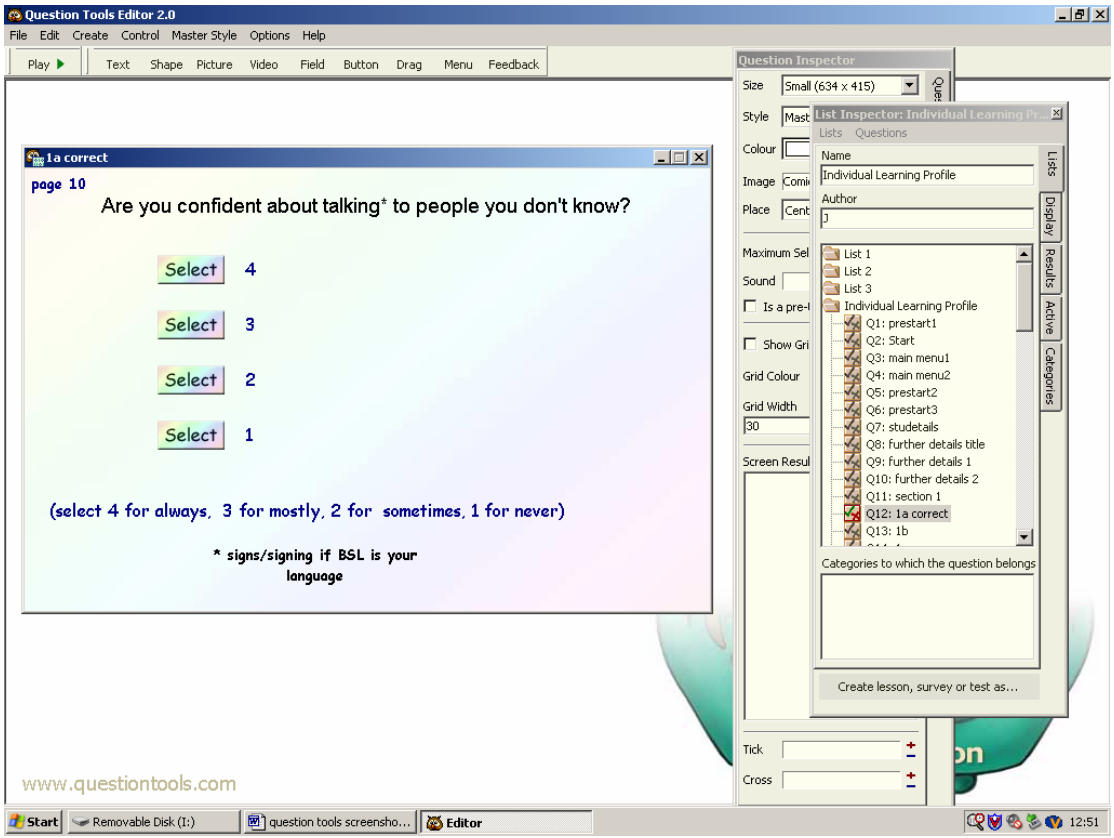


Fig 6.22 First question in the ‘Speaking & listening’ section.

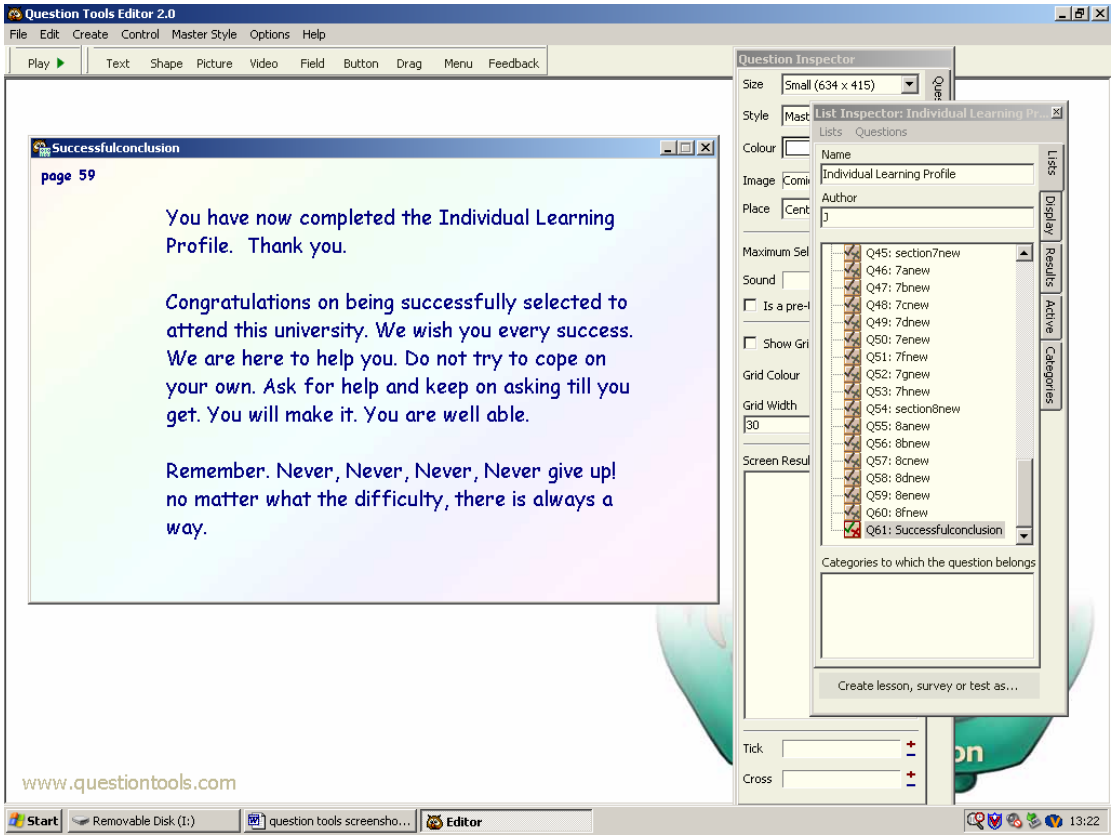
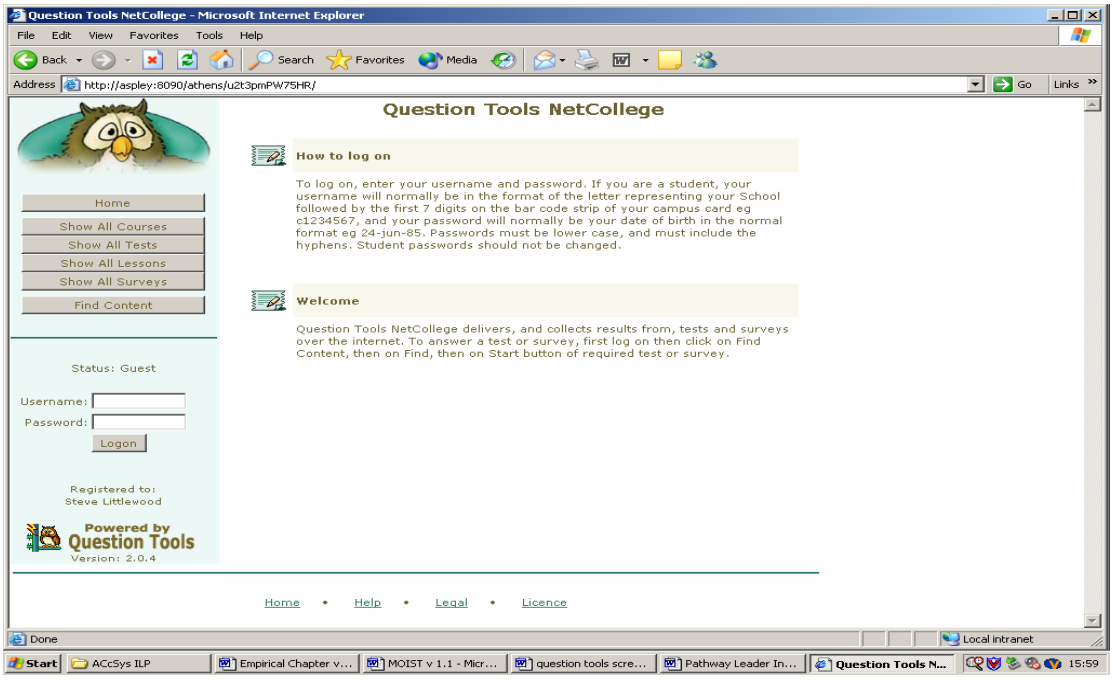


Fig 6.23 End screen of ILP providing some motivation and encouragement for the users

Instruction Sheet
[i]ACcSys Individual Learning Profile Electronic System

- Open Internet Explorer

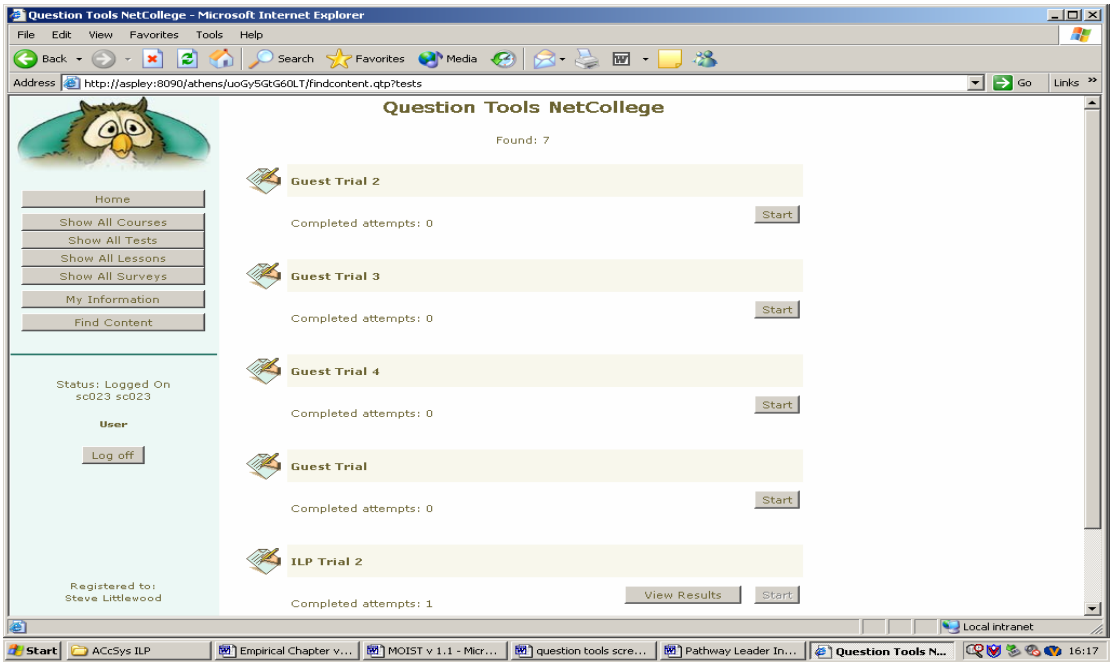
- Set all computers in the designated lab to the URL http://aspley:8090



- Each student should login with the assigned username and password details

- After login they will see buttons to the left and a blank area to the right of the screen. It will say ‘status: Logged on’

- Use mouse to click on and select the ‘**Show all Tests’** button on the left



- There will now be a choice of tests
- Go to the one called '**ILP Trial 2'**
- Use mouse to click on and select the **'start'** button to the right.
- The Electronic ILP questionnaire will appear
- Follow the instructions and do the test
- Use the '**Next'** and '**Back'** navigation buttons or slider to go forwards or backwards
- Select only one (1) answer for each page.
- To change an answer to a question, simply select the desired answer.
- Click on the **'Finish'** button once answers are completed
- Click '**Ok'** if finished or '**Cancel'** if not finished
- Ignore the results on the page that comes next. These are not the true results and will not be used.
- Click the '**Close'** button
- Use mouse to click the '**Log off'** button. Click '**OK'**
- Results are stored and will be retrieved later by authorised personnel

6.8 Empirical Study Phase 5 – From Implementation to Testing

User testing is of vital importance to the development process and the quality of the final product. This testing should occur throughout the life of the design and development process. A focus group is selected. These are a group of randomly selected people who represent the target audience. This selection is extremely necessary as it can save hundreds of production hours later on. The designers are the ones who make the structural and user interface decisions. As they become more intimately involved with the project, it very easy for them to lose objectivity and not see obvious flaws. That is why user testing brings a fresh and more accurate and balanced perspective to the whole design and development process. They help to determine whether the product is understandable to a mass audience.

6.8.1 User Testing I

A group of new students earmarked for entry to the university in September 2004 were used to test the system. These were 70 students on a 4 weeks Mathematics bridging course at the university. The aim of the course is to correct math deficiencies and bring the students up to a suitable level of competence in mathematics. They were an ideal test group for the research as they provided the profile of students who will officially use the system in September.

Eight (8) students participated in the first testing session. The participants were asked to complete the ILP questionnaire electronically. The results were exported in comma separated value (CSV) format to a MySQL database and the relevant values and lists were extracted and generated by PHP code. They enabled us to see how the system worked, to evaluate it and to eradicate bottlenecks for smoother working.
The following heuristics were designed to test the electronic system. User friendliness of the system, quality of the interface, colour scheme appropriateness, error handling, navigatability. An evaluation questionnaire was designed and each student who tested the system was asked to evaluate the system on the criteria given and to give general feedback on how it could be improved. They were not given major instructions as to how it worked.

6.8.2 First Evaluation Questionnaire for the ILP Electronic System

Question one:

Was the system easy to use?

Yes[7] No[0] it was not too easy or difficult[1]

Question 2

How user friendly was it?

Very[7] Not very[0] horrible to navigate[1]

Question three

Was the colour scheme appropriate?

It was okay[7] too bright[1] too pale[0] should be changed[0]

Question four

How were the questions?

Sensible[8] silly[0] need reworking[0]

Question five

Could you go from one page to another easily?

Yes[7] No[1] I got stuck[0]

Question six

Did you understand what to do when using the system?

Yes[6] No[0] Sometimes[2]

Question seven

Was the font size okay?

Just right[7] Too large[1] too small[0]

How could the entire system be improved?

- It does not need many changes
- Make the navigation system better and use better colours

The suggestions were then implemented before the next testing session.

6.8.3: User Testing II

A presentation was scheduled for senior lecturers and Heads of departments from both the Schools of Computing and Engineering. Five (5) senior lecturers were represented from the major departments of both schools. The aim of the presentation was to demonstrate the electronic system and ensure their competency in the same. This was to conform to the new format where each pathway leader was responsible for his or her set of students and therefore would administer the electronic ILP. The benefits were many. It would break up the mammoth task of administering the electronic ILP into manageable portions as delegation occurred. It will help the academic support tutor to more closely focus on the task at hand which was to provide the actual support to students, instead of spending months on administration of paper based results. The pathway leaders would also have a head start on the educational level of their students and could tailor their courses accordingly.

6.8.4 Second evaluation

- The main corrections were syntactic and semantic in nature
- One complaint centred around the selection and feedback feature as there was no feedback on the button selected and there was no automatic advancing to the next question.
- One comment was that it was a very good system and that it is more than they had expected so quickly and the important thing is that it works and will make a difference in the academic support process
- One question asked was if students could print off their own copies in order to monitor their own progress and keep it in their Personal Development Portfolio (PDP) files

The first comment is easily amendable, but the second weakness is central to QT and is therefore not easily done immediately. The benefit to students' Personal Development portfolio (PDP) is an indirect, but very important one. The ACcSys was not intended to be used for that purpose, but it has proved to be an additional benefit that will help the academic schools that use it. The PDP is being championed by senior management to help students make a more successful and smoother transition from academia to the workplace.

6.8.5 Live Run of the ACcSYs Electronic ILP during Induction Week

For induction week, the ACcSys ILP faced its biggest test. The students were timetabled to maximise the optimal lab facilities. They were divided into pathways and pathway leaders were assigned to man the session.

Date	Student ID Code	Pathway
21st Sept, 2004	Sc024 – sc063	Computer Games Programming
22nd Sept, 2004	Sc064 – sc084	HND BIT
22nd Sept, 2004	Sc086 – sc100	Electrical and Electronic Engineering
23rd Sept, 2004	Sc101 - Sc191	Music and Technology Engineering
24th Sept, 2004	Sc192 – sc231	Computing and Mathematics

Table 6.7: School Induction Timetable

6.8.6 Problems encountered in ACcSys's maiden run

The first problem surfaced. Some of the pathway leaders invited to be trained in operating ACcSys, were not able to come to the training session, owing to prior commitments. Available personnel were deployed to alleviate this. Another problem was that the central student details that should have been provided for prior uploading to the system could not be provided by university central registry as student registration was still in progress. Dummy passwords were then generated for all the Computing and Engineering students. The other glitch came when students who had their ID cards could not log in to the main system as it was still officially registration week. Consequently, their student numbers were still being processed by registry. To solve that problem temporary logins were assigned to each student. This led to yet another problem as the traffic proved too much for the temporary logins to handle. This was solved by administering the diagnostic tests using smaller batches of students.

Another problem cropped up with the server where the Question Tools based ILP resided. Owing to the traffic overload, accessing QT proved difficult at times. Eventually this was solved by a script being written to get the server started again. Operations improved as the induction week progressed. One recommendation for future administration of the ACcSys is to wait till the university's official registration week is over. This would eliminate most of the bottlenecks experienced in the first live run, as student data would be fully processed and available.

The students completed the questions electronically. The results were then automatically calculated by the system. At the end of induction week, the results

were generated into comma-separated text and exported to the already existing MySQL database. PHP code was run to extract the pertinent details and reports were generated. In less than one and a half hours after exporting the results to the back end database system, the reports were ready. What took more than four (4) months to accomplish in the previous academic year happened in less than two hours. The reports were available for viewing. The reports were disseminated to some lecturers that same day and the subsequent academic day. The academic schools who opted not to take the electronic ILP route still had not processed their data. They were still in the paper based format waiting to be processed. The ACcSys system works and works very well. It solves a grave problem and enables the Academic skills tutor to get down to the raison d'etre of identifying and supporting 'at risk' students.

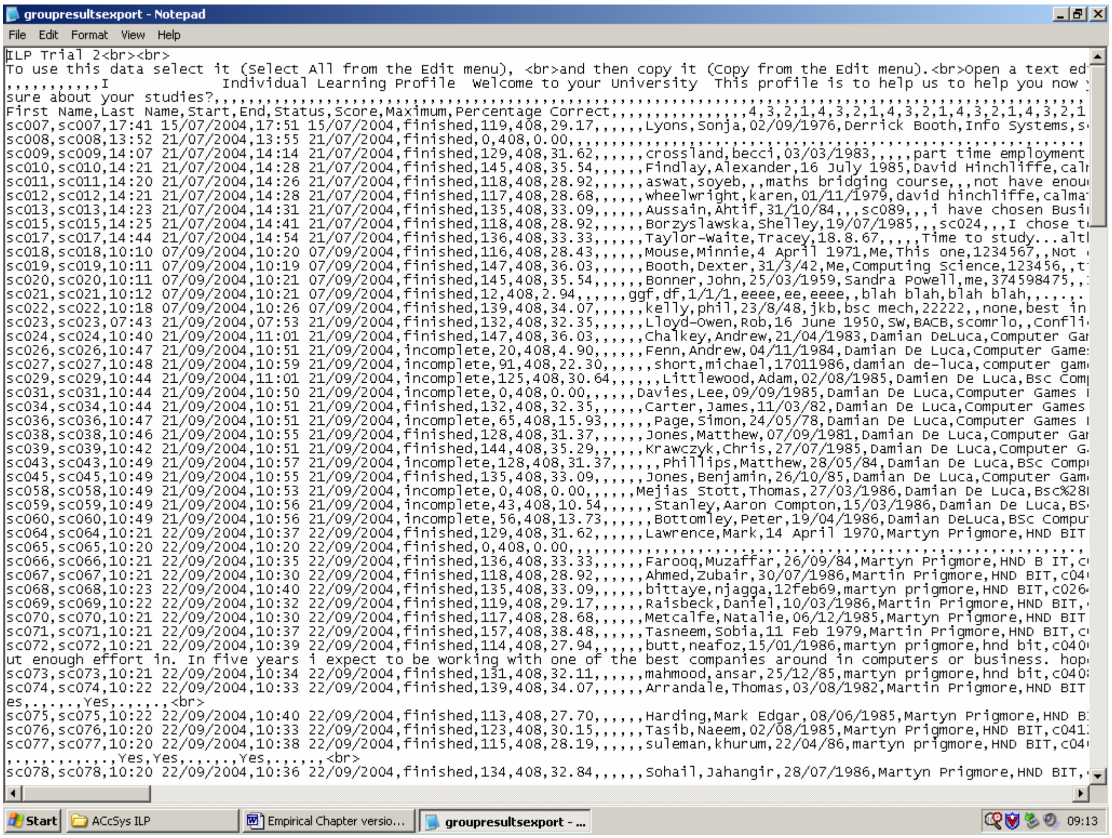


Fig 6.24 Screenshot of comma separated format (csv) data generated by the ACcSys

6.8.7 Results of Using the MoIST Method

One of the major benefits that resulted from using the MoIST method in the Academic Support Process was an improved user requirements definition. This was vital to the successful implementation of the system. The effectiveness of the MoIST method was mainly evaluated against the criterion of whether or not, there had been an effective intervention. Significant lessons were gained from the experience of using the MoIST method in the academic support process.

6.9 Conclusion

The outcomes of the intervention in the academic skills support development effort clearly demonstrates that synergy occurs when different methodologies from different disciplines are employed in information systems development (Xu, 1995). An IS project concerns an interplay of human, organization and technical factors which are not easily separated (Walsham et al, 1988). SSM and OOA are viewed not as self-contained methodologies to IS development, but as approaches which can work together. A complementary application of both methodologies would assist systems developers in minimising the many failure cases of I S (Lai, 2000).

MoIST's effectiveness is shown in the empirical data above. To further prove its usefulness and relevance, it is also shown when it is used in a separate environment. This situation is in the postgraduate project process of the School of Computing and Engineering.

Chapter 7. Study of the Postgraduate Project Process in the School of Computing and Engineering

7.1 Introduction

The research empirical data showcased here provides evidence that MoIST is effective in a real world situation. To further underline its efficacy. MoIST will be used in yet another real world context. This is in the Postgraduate Project Process in the School of Computing and Engineering.

The postgraduate students of the school of Computing and Engineering do the major work on their projects when taught classes have officially ended. This usually runs from end of May to end of August of an academic year. The entire postgraduate process involves many team players. Even though the MSc students have a vital role to play, there are many behind the scenes persons who work assiduously to ensure that the entire process flows smoothly.

7.2 Description of current system

To do an excellent postgraduate project is quite an involved process. It is not merely about developing a system to solve problems or to enhance the operations in an application area. It consists of a more in-depth process. 'Academic projects should provide evidence of a much deeper understanding of what you are doing. They require some form of justification and contextualisation. You are not expected to do merely what you are told to do, but you are expected to develop your own thoughts, arguments, ideas and concepts. You are expected to question things and look at things in new ways and from new angles (Dawson, 2000, p1).

There are currently 4 pathways of the taught MSc postgraduate programme offered in the School of Computing. They are Internet Application and Development. There is also MSc in Software Development, MSc in Information Systems and MSc in Interactive Multimedia. The MSc Internet Application Development is geared towards persons who wish to make internet development a career. The MSc software development targets those who desire to be designers and programmers. The MSc Information Systems attracts those who are interested in Systems Analysis and Management. The MSc Interactive Multimedia is for persons who want to work in cutting edge multimedia environments. With the exception of Interactive Multimedia, the first semester for the four pathways offers the same modules. Specialisation occurs in the second semester.

There is a fulltime and a part-time mode. The fulltime mode is for one academic year in duration and the part-time mode for two years. The majority of students successfully complete all modules of the two semesters of their MSc Stream. If students fail any of the taught modules in their pathway, they have the option to go on the individualised pathway, retake any module they failed and continue with the project on a part-time basis. Each MSc is considered a pathway and is assigned a pathway leader. The major stakeholders in the postgraduate project process are the pathway leaders, several academic supervisors who have direct and one to one contact with the project students, the project tutor and there is also a postgraduate scheme administrator responsible for managing student details. These project stakeholders meet at the end of every academic year as a Pathway Assessment Board. Here external examiners are called in to inspect procedures and results and award degrees.

The MSc student completes two 15 week semesters of taught modules. On successful completion of all modules, the student is ready to start the project. The student information is passed to the postgraduate scheme administrator by the Pathway Leader. This ensures that the student is registered and is 'live' on the system. This information is vital to the registry and finance departments. It is imperative that they know which students are enrolled at any given time. After enrolment, the student then has access to the intranet (currently Blackboard version 6). They go to the projects module and retrieve information about past projects and supervisors and try to match them with their interests. Students generally are required to identify and delineate a project and obtain agreement with stakeholders. Specifically they find external clients and contact the relevant academic supervisor to ascertain their availability and willingness to supervise them. The students subsequently meet with the Project Tutor who disseminates further relevant information and tries to ensure that students know what is expected of them. Meetings with external clients are arranged. Terms of reference need to be completed and handed in to the office before the project will be deemed to have officially started. Arrangements are then made with the appointed academic supervisor for regular meetings. The student is expected to plan, manage and execute the project using skills gained in taught pathway sessions. The academic supervisor's role is to advise the students on the project from an academic standpoint. In a more strategic sense, the supervisor helps the research students develop into individuals who think and behave as academic researchers in their field of study.

7.3 Application of MoIST to the Postgraduate Project Process

Empirical Study Phase 1 – Analysis and 'finding out'

This phase explores the process involved in doing postgraduate MSc projects in the School of Computing and Engineering. The MoIST method was applied to the situation. The relevant data were gathered, collated, analysed and the outcomes mapped to the design phase.

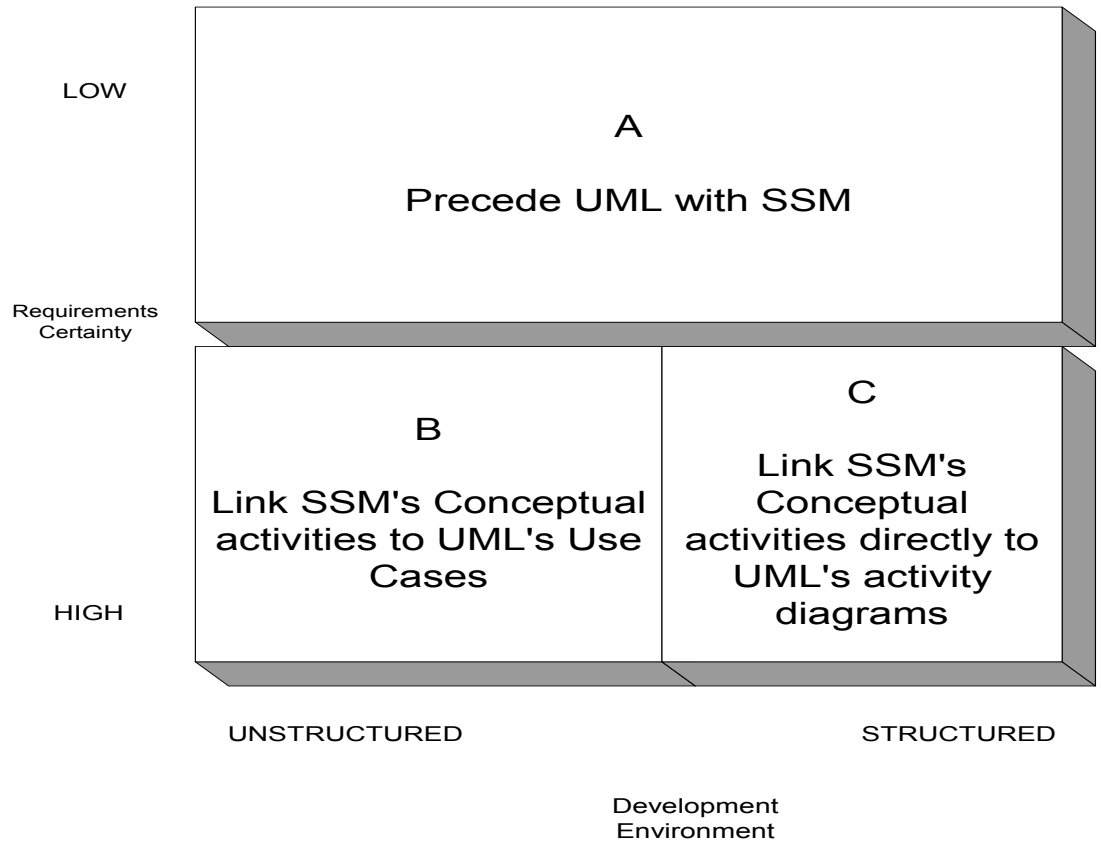


Fig 7.1 MoIST model

7.3.1 Using MoIST Project Option Selector Tool to select the best project option

The characteristics of the project were analysed using the MoIST Project Option Selector Tool in order to make the most informed decision. The characteristics of the Postgraduate Project Process Project were:

- Organizational changes are likely
- The proposed system is to enhance an existing system
- Development environment is unstructured
- Requirements are not relatively clear from the outset.

MoIST's Project Option Selection Tool (MoPros)

max. 25 points max. 25 points max. 25 points

max. 25 points

Project Options	Types of users	Developers' skillsets	Organizational environment	General characteristics	Total
A	Users are a bit unsettled as they are experiencing organizational changes 15 points	Requirements at this point are not relatively clear to the development team 15 points	Development environment unstructured 20 points	Proposed system is to replace or enhance an existing system 25 points	75
B	Users uncertain about the need for the proposed system while others are more willing to be associated with it. 15 points	Requirements known at this point are relatively clear to 80% of the development team 5 points	Development environment has pockets of structured and unstructuredness. 10 points	Conflicting interests and the proposed system might cross functional borders 4 points	34
C	Users open to the new system 0 points	Requirements known at this information are very clear to 90% of the development team 0 points	Development environment is quite structured 0 points	Environment is relatively contention free 20 points	20

Table 7.1: MoIST's Project Option Selection Tool (MoPros)

Using the MoIST Project Option Selector tool, it was found that the project requirements most closely matched project option A, so **Option A** was chosen.

MoIST Project Option A -Precede UML with SSM

Status: Requirements Certainty (Low) + Development Environment (Unstructured or Structured)

MoIST Option A's Activities
11. Requirements for computer-based information system 12. construct rich picture 13. develop relevant issue-based and primary task root definitions and conceptual models 14. derive consensus primary task model and information categories 15. formulate the recommendations for information system design

7.3.2 Mode of data gathering and interview data

Three categories of relevant stakeholders in the Postgraduate Process were interviewed. These were Project Tutor, Postgraduate Scheme Administrator and students. While the research was in progress, a new project tutor and a new Postgraduate Scheme Administrator were appointed. These two new persons were subsequently interviewed. This was no inconvenience to the research, but a welcome addition. It added to the depth and accuracy of the research as it provided more opinions and ideas and opinions from the same vantage point. Students from both the part-time and full-time mode were also interviewed.

Some interviews were recorded on audio tape and some were handwritten on notepad; but all were transcribed to increase the accuracy of the opinions recorded. Soft Systems Methodology (SSM) was then applied. Usually in SSM it is recommended that the Soft Systems methodologist should organize a meeting of the stakeholders in order to agree a primary task model. SSM however is flexible and it was thought not necessary to hold such a meeting at this point.

Soft Systems Methodology (SSM) is a study made popular by Professor Peter Checkland formerly of Lancaster University. It is a methodology that seeks to encourage learning and bring understanding and change to unstructured situations in organizations. It purports to unearth the problems and recommend change when there seems to be a maze of unresolved issues. It uses constructs called human activity systems to model and represent the situation to bring more clarity. Relevant systems or activities are then deduced from the models and compared with the real

world. It is thought to be then easier to see and implement changes that are systemically desirable and culturally feasible in the context of each particular organization. SSM consists of several phases. The first is the finding out phase where a rich picture is constructed. This depicts in diagrammatic and pictorial form, the methodologist's view of the actual goings on within the organization. Any conflicts, hidden agenda and power plays are noted and recorded. This is what depicts the reality of the situation.

Analyses 1, 2 and 3 are then conducted. Analysis is done of the intervention into the situation, analysis is done of the social construction of the organization and looks at the social roles, behavioural norms of the role holders and values that measure role performances of the parties concerned as to whether they are good or bad. Analysis is also done of the balance of power in the organisation and how it is preserved, grabbed or passed on. Relevant systems or definite categories of activities are identified. For each of these systems or categories, succinct definitions called Root Definitions are summarised. These root definitions are evaluated according to the CATWOE template which has the textual formulae of 'a system to do X by means of Y'. Activities are then deduced from the root definition and are arranged in some logical order to form conceptual models. These models are abstract and do not represent the reality of the organisation. In order to see where change is needed, the disparity between the abstract conceptual models and the stark reality of the rich picture are contrasted. Areas that need change are then more easily thrown into relief and stand out. These become the recommended change areas. Once these change areas are systemically desirable and culturally feasible to the people in the organization, it can be accepted. SSM is quite flexible and depending on the nature of the project, some of the phases can be left out or done in varying order. There is no definite cut off point for SSM for it is a learning system. Its end is subject to the will of the methodologist and to other pressing environmental constraints.

7.4 Problem Situation

Interviewing the various stakeholders shed light on difficulties present in the current postgraduate project support system. One major problem that students face is that external project clients sometimes change the project 'goalposts' after terms have been agreed. This is sometimes due to not having enough finances to execute what they really wanted from the project. At other times, it is because the company might have been taken over by a larger concern. Yet another issue that needs to be addressed is the fact that computer science students need better dissertation writing up skills. Students struggle with referencing and research methods skills. Some students have no idea how to start the project. It was thought that MSc Software Development students tended to do better as it is mandated that they create a software product in order to secure at minimum, a pass in their academic programme. MSc Information Systems students on the other hand need more methodologically based arguments in order to write up their dissertation. They also need greater analytical skills to give their projects more credence. MSc Interactive Multimedia students tend to do very well. Of the 19 students who did projects for Interactive Multimedia in 2002, only 3 did not graduate. The research noted that Interactive Multimedia is the only pathway that does group projects. This raised some questions as to whether this factor had any bearing on the grades being produced. It is beyond the scope of this particular research paper, but will be noted for future work. The Postgraduate Scheme Administrator currently uses the Student Programme Route Database (SPR). This is not able to address all of the issues concerning postgraduate projects, so a shared database that is specific to projects and that gives timely reminders would be useful. Another identified issue was that the external examiners need to know exactly how the marks were arrived at. Sometimes registry or finance departments would contact the computing administrative office to ascertain student's current status and to see whether or not they should be billed for an academic year. The students also need to enrol if they go beyond specified finish date as they need to be 'live' on the systems so grades can be entered and the computing and intranet resources made available to them. The administrator always has to get current student information from a module tutor and also has to query students' start and expected completion date. If there is no module tutor available, this results in bottlenecks in both registry's and finance's efficiency. This is bad for business. In a climate where inefficiency spells loss of money, this state of events is untenable.

Additional Issues

- dedicated monitoring of part-time MSc students is needed. Students transfer or are transferred to part-time mode for varying reasons. They have 15 months to complete the project. It is increasingly becoming a problem that they are failing to submit on time or on an even larger scale, failing to submit their projects at all.
- The very short time for doing full-time project work is only 2 months and students have to exercise very good time management skills. This means that it is imperative that the project management team ensures that the support and control systems are very good.
- Students need to learn how to manage their relationships with their clients

Application of MoIST option 'A' using SSM

These are the analyses done on the basis of the interview and observation data gathered.

Analysis One - Analysis of the intervention in the situation

Clients (Who caused the study To take place)	**MSc students, project tutor**
Would-be problem solvers (who conducts the study)	**SSM Methodologist, project tutor**
Problem Owners (client + people with an interest in the situation)	**MSc student, client, supervisor** **Project management team**

Table 7.2: Analysis of the Intervention of the Postgraduate Project Process

Analysis Two

This looks at the problem situation as a social system and helps to determine cultural feasibility of any changes to be recommended. It examines the social roles that are significant in the situation, the behavioural norms of the role holders and the values that measure role performance as good and bad.

Social Roles	Behavioural Norms of Role holders	Values that measure role performances as good or bad
Student	Identifies and delineates a project and obtains agreement with stakeholders. Plans, manages and executes project using skills gained in taught pathway sessions	Registers for projects module and identifies projects in which interested. Accepts clients and projects. **GOOD** Submits and delivers project and dissertation to university examiners. **GOOD** Not maintaining regular work pattern and not being honest when reporting progress. **BAD**
Academic Supervisor	Advises students on the project from an academic standpoint. Helps to clarify references and act as a sounding board for ideas.	Reads students' work well in advance. **GOOD** Being available when needed. **GOOD** Being unfriendly, closed and unsupportive **BAD**
External client	Project proposals submitted by potential client	Invite student for discussion & interview **GOOD** Provide feedback to student & project tutor on project quality **GOOD** Changes initial terms of project agreement midway through project **BAD**
Postgraduate Scheme administrator	At start and end of project responsible for accurate input of student details & grades into database	Ensures that students are registered in order to gain intranet access. **GOOD** Sits on pathway assessment board meetings. **GOOD** Inaccurate input of student data **BAD**
Project Tutor	Oversees all project aspects of all MSc pathways	Allocates academic supervisor to student **GOOD** Arbitrate disputes **GOOD** Not available to define and vet submitted proposals **BAD**

Pathway Leader	Monitors overall progress of students on the pathway	Liaises with project tutor and supervisors **GOOD** Does not have a handle on what is happening with students in the pathway **BAD**

Table 7.3: Analysis 2 of the Postgraduate Project Process

Analysis three

Examines the politics and power distribution in the organization

Disposition of Power

Three players in the postgraduate project process hold the most power and these are firstly the MSc Student, secondly the Client and thirdly the Academic Supervisor. The student is the one who has to do the work. In a sense, a less than ideal client or supervisor should not prevent a student from achieving the research goal at hand. The client was second in the power stakes because they are the key to the research problem which is the fulcrum on which the entire research project is based. The academic supervisor's expertise in the research area and ability to supervise and relate well to students is also a major factor in the research process.

Nature of Power

The Project Tutor and Pathway Leader are not line management roles, so there is not that sense of hierarchy or of 'lording it over each other'. Instead there is the sense of team effort to ensure that the postgraduate project process should result in successful research projects.

The external client, whilst outside of the organisation can ruin a student's chances if for instance, they change the terms of agreement almost at the end of a project. That is vicarious power and it is quite detrimental if the client does not exercise it prudently and reasonably. The academic supervisor within the organisation has the power to demean, demoralise and demotivate a student. If there are personality clashes, this can adversely affect the success of the student research project.

The postgraduate scheme administrator if inefficient can cause serious bottlenecks in the system. If incorrect grades are entered and sent to the pathway assessment board, this has serious repercussions for the student, if they are not proactive enough to investigate. Efficiency and support in this role is quite important to coordinating the overall efforts of the entire postgraduate project management team and is quite a key area.

Rich picture

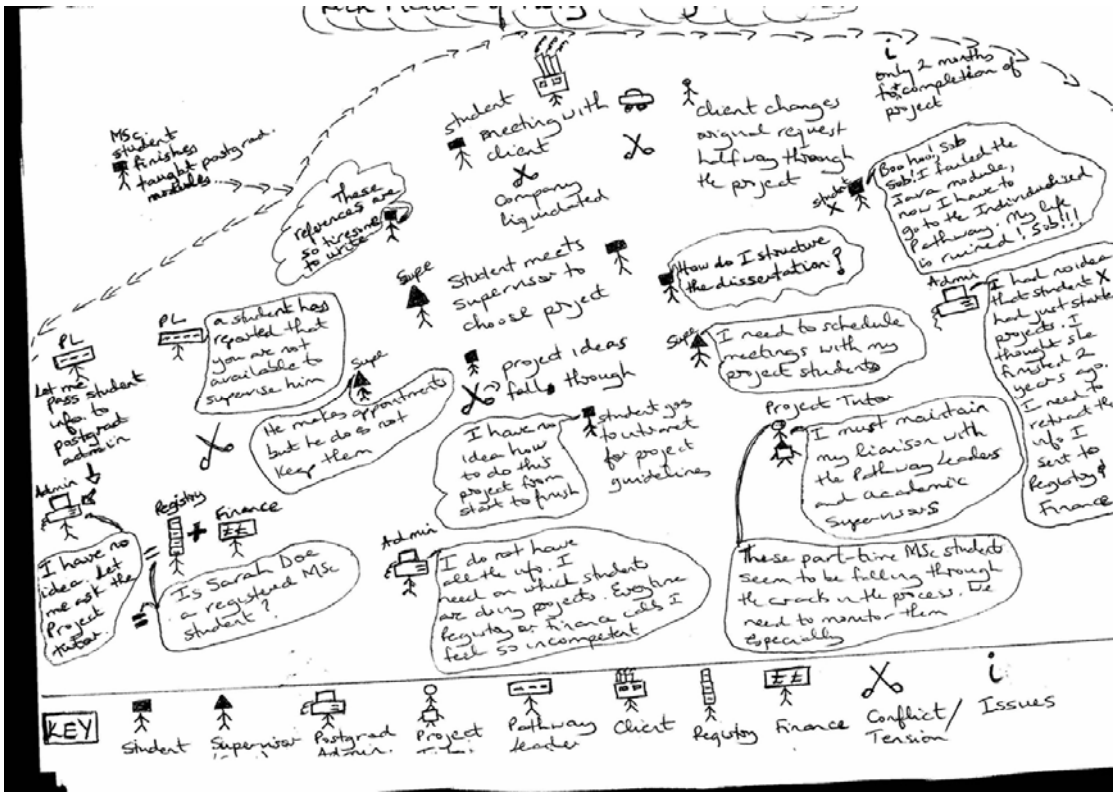


Figure 7.2: Rich picture of Postgraduate Project Situation.

Formulating Root Definitions(RDs)

Primary task based Root Definitions (related to basic set of tasks)

"a system to do a planned research project acceptable to a university by means of literature search, submission of project proposal and application of relevant research methods to solve a suitable research problem in order to achieve a Masters degree"

Table 7.3: Root Definition of Postgraduate Process

CATWOE analysis

mnemonic which helps guide and ensure the well-formedness of root definitions

C	**Customer**	MSc Student
A	Actor	Project Management team, MSc student
T	Transformation	research problem --→ problem solved
W	Weltanschauung	It is important to follow a scheduled process to achieve a masters degree
O	Owner	School postgraduate project management team
E	Environment	university requirements, academic supervisor

Issue based Root Definitions

"a system to provide understanding on how to write a good proposal by examining guidelines and past examples, in order to develop a manageable research project"

"a system to provide peer support by maintaining regular contact with other MSc students in order to maximise group learning"

"a system to provide effective support from supervisors, by managing the student/supervisor relationship in order to achieve maximum benefit"

Primary Task Conceptual Model

Owing to the flexibility of SSM, it was not thought necessary to show the client the conceptual model before coming to a decision.

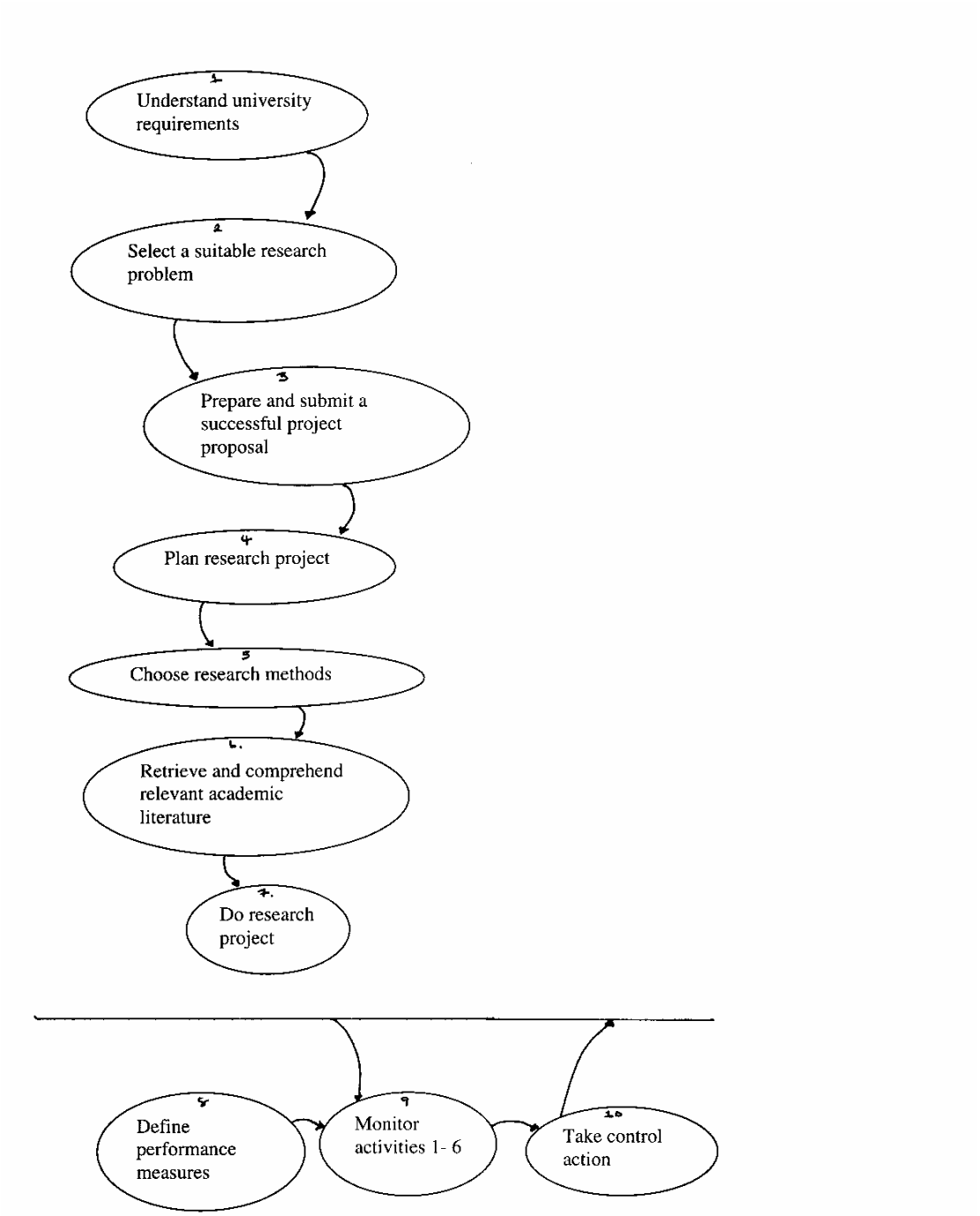


Figure 7.3: Conceptual Model of Postgraduate Project Process

Measures of Performance

These are criteria used to evaluate the correctness of the conceptual model. They are popularly known as the 3 E's. In actuality they are really five measures of performance

Effectiveness

Does the planned research project increase the likelihood of achieving the Masters degree?

Efficacy

Will doing literature search, submitting project proposal and applying research methods make the masters degree achievable?

Efficiency

If the resources expended in achieving the research problem solved was not detrimental cost to life, family and health, it is worthwhile being expended for that aim. Financial resources can always be recouped and achieving the MSc can be perceived as an investment in a better future.

Comparison Phase

Conceptual	Reality	Implications
1. understand university requirements	There is extensive literature on the intranet, library and in hard copy format. The academic skills unit are compiling a definitive university reference handbook.	Students need to utilise the available resources and ask for help if in doubt
2. Select a suitable research problem	Some students are not certain what research problem to choose. There is no formal list of past research projects and related supervisors and no list of current research areas and supervisors.	There is past research projects repository facility for undergraduates called POD, but there also need to be one for postgraduates. Some information on project areas and supervisors does exist on the intranet, but the need exists for a more formal compilation.
Prepare and submit a successful project proposal	There is reasonably adequate provision for this with MSc students. Resources are available on the internet and actual old copies may be had from academic supervisors	A template for well structured proposal could be made available to students
5. Choose Research methods	This area needs some help. Students especially non-social science students are not formally taught research methods.	There needs to be some concerted effort made to teach students research methods and how to conduct academic research to help them in the writing up stage
6. Retrieve and comprehend relevant academic literature	There are more than adequate resources for garnering relevant literature area.	Students however usually need help with the techniques of writing a literature review. They need assistance to determine what to let stay in and what to leave out. The supervisor's assistance is crucial in this
4. Plan research project	This involves time management and the onus is on the student to do a Gantt chart and plan milestones carefully and on the supervisor to ensure that this	Some sort of technique or other help in this area can be provided by supervisors for students

	is being adhered to	
7. Do research project	All the above mentioned areas are vital to the actual doing of the research projects	If change is applied to these areas, the doing of the research project will be less difficult
8,9, 10 Monitoring and Control of system to do a research project	Any monitoring and control of student projects done is left to the discretion of the individual supervisor. There is no standard monitoring and control process in place	There needs to be some sort of system in place to check student progress and to support the administration of the entire projects process

Table 7.4: Comparison Phase of the Postgraduate Project Process

Defining changes

The changes proposed below are derived from a comparison of the conceptual model with the problem situation and are perceived to fulfil the twin criteria of 'systemically desirable' and 'culturally feasible' put forward by Checkland(1981), Checkland & Scholes(1990) who said, 'they are systemically desirable if these relevant systems are perceived to be truly relevant'.

These changes require no extensive resources to implement and will be welcomed within the school. This therefore qualifies the defined changes as culturally feasible. Checkland's justification for this criteria is that regular 'hard' systems engineering does not usually check whether a system will fit into the context of an organization. Consequently once the engineered product is technically sound, it is automatically assumed by the developers to be 'systemically feasible' with scant regard for how the people that drive the politics, that is the users will receive it in their everyday work mode. SSM seeks to take the users in an organization and their cultural context into consideration, before recommending changes. If SSM proposes a change that will be systemically sound, but culturally infeasible, it is automatically scrapped and another alternative sought.

Conceptual Activity	Proposed Change
1. understand university requirements 6. retrieve and comprehend relevant academic literature	Run a short intensive course in how to do research in computing. This could consist of two project seminars run at the beginning and midway through the project. Topics could involve 'writing references the university way', doing a literature review. Students could ask questions and get immediate feedback.
4. Plan research project	Provide a detailed project life cycle complete with suggested durations and milestones to help the MSc student better plan the research projects. This will also give the supervisor a template for progress monitoring
7. do research project	Organize peer support by assigning students on the same pathway to small groups. They could meet at least twice during the project and utilise the intranet for discussion at agreed times when necessary. This would be especially useful for part-time MSc students as it provides accountability and indirect monitoring of their research project progress
8, 9, 10 Monitoring and control activities	In order to monitor the system of doing a research project. The monitoring and control activities 8 -10 will be expanded to become a system. This monitoring system will track the progress of the students' research project and will provide support to the process. The monitoring system is therefore a tracking system of sorts.
8, 9, 10 Monitoring and control activities	Construct a standardized feedback form for supervisors. This could be modelled on the existing undergraduate feedback forms in the Open Learning Centre. This would provide a recorded history of marks and enable external examiners to more clearly and accurately see how marks and grades are arrived at.

Table 7.5: Proposed Changes to the Postgraduate Project Process

Expanding the Original Conceptual Model

The monitoring and control activities 8, 9 and 10 of the postgraduate project process will be expanded. This will have a tracking function and will monitor whether the research process is being carried out according to the measures of performance defined.

A relevant system or root definition will be formulated for this monitoring and control system and will become a subsystem to the main research project system. This tracking subsystem will also have its own monitoring and control activities.

Root Definitions for expanded monitoring and control system

Primary task based Root Definitions (related to basic set of tasks)

'a system owned and operated by the Postgraduate Project Management team of the School of Computing **to monitor and track the progress and performance of postgraduate project students** in order to ascertain the current status of students at any given point in time; alerting the project management team if necessary to the need for timely execution of control action in the event of potential hindrances, bottlenecks or threats that could lead to a drop in student performance thereby ultimately improving the quality of dissertations submitted and increasing the number of successful postgraduate project students'

Table 7.6: Root Definition of expanded Postgraduate Project Process

CATWOE	
C – Customer	postgraduate scheme administrator, MSc project management team, students
A – Actor	MSc project management team – project tutor, administrator, pathway leaders, supervisors
T – Transformation	no means of monitoring students-> means of monitoring students
W – Weltanschauung	monitoring the students' research projects help to increase the number of successful students
O – owner	School of Computing postgraduate project team
E – Environment	resources, university requirements

Tracking System Conceptual Model

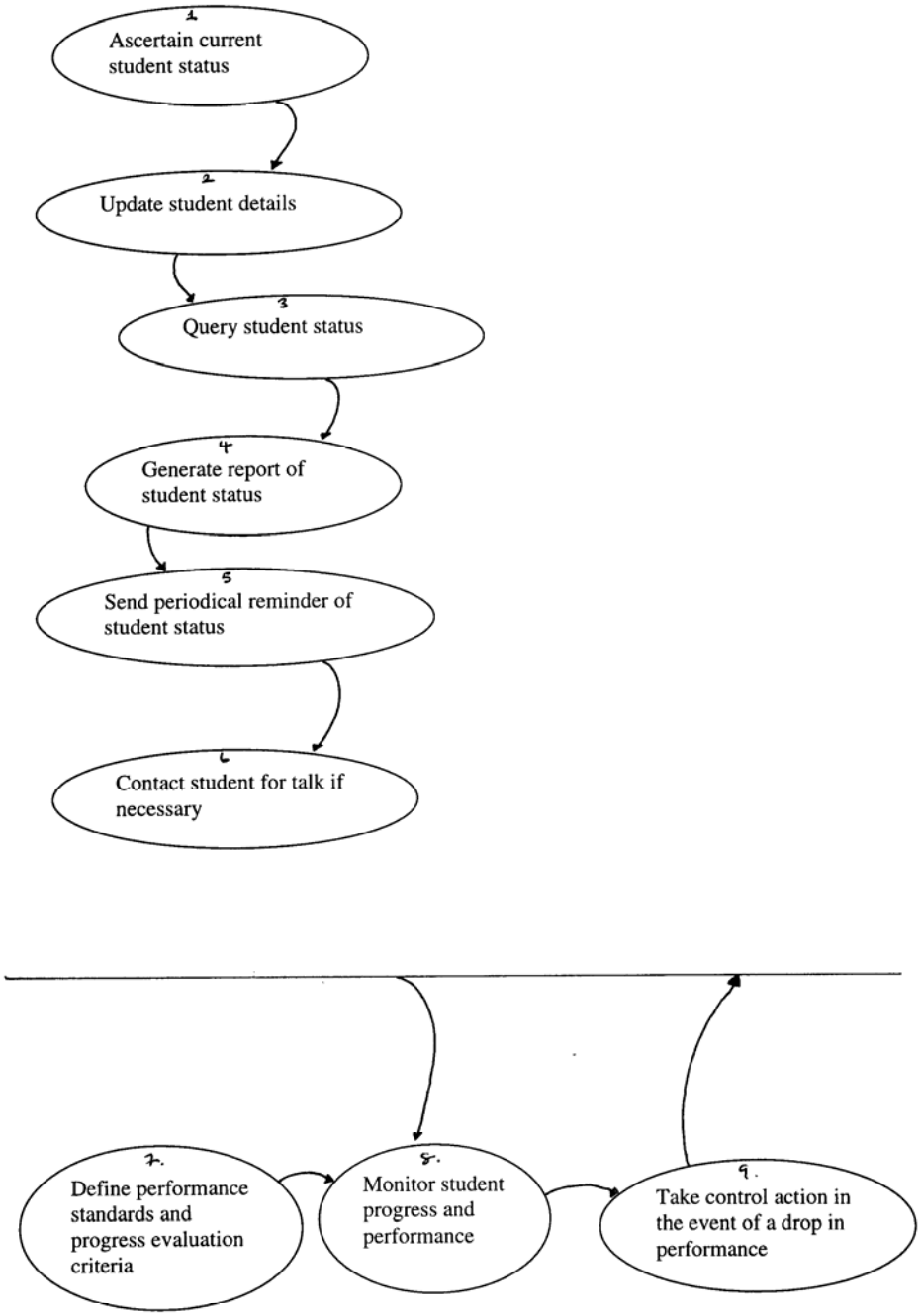


Figure 7.5: Conceptual Model of proposed tracking system for PPSS

This tracking system above provides monitoring and control functions for the original conceptual model. This gives rise to a nested conceptual model as shown below.

Amplified Conceptual Model

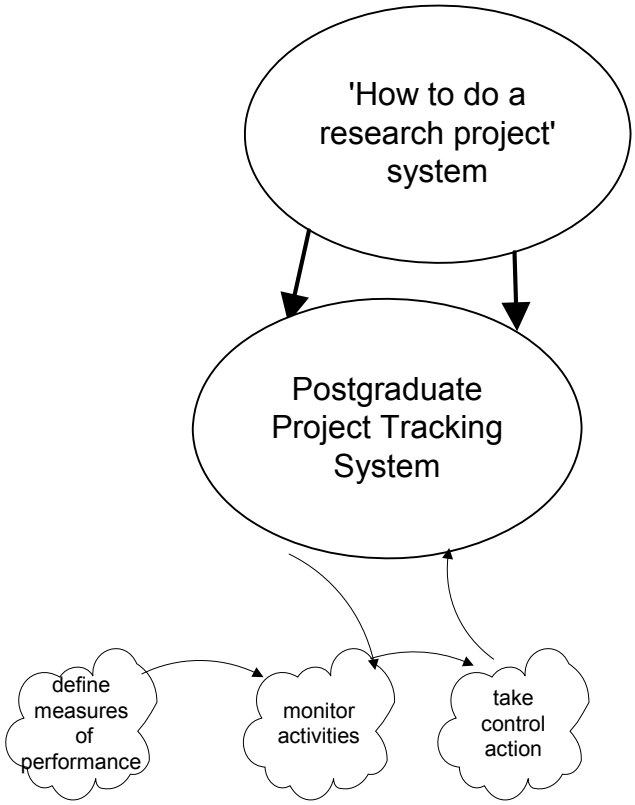


Figure 7.4: Another Conceptual Model for Postgraduate Project Process

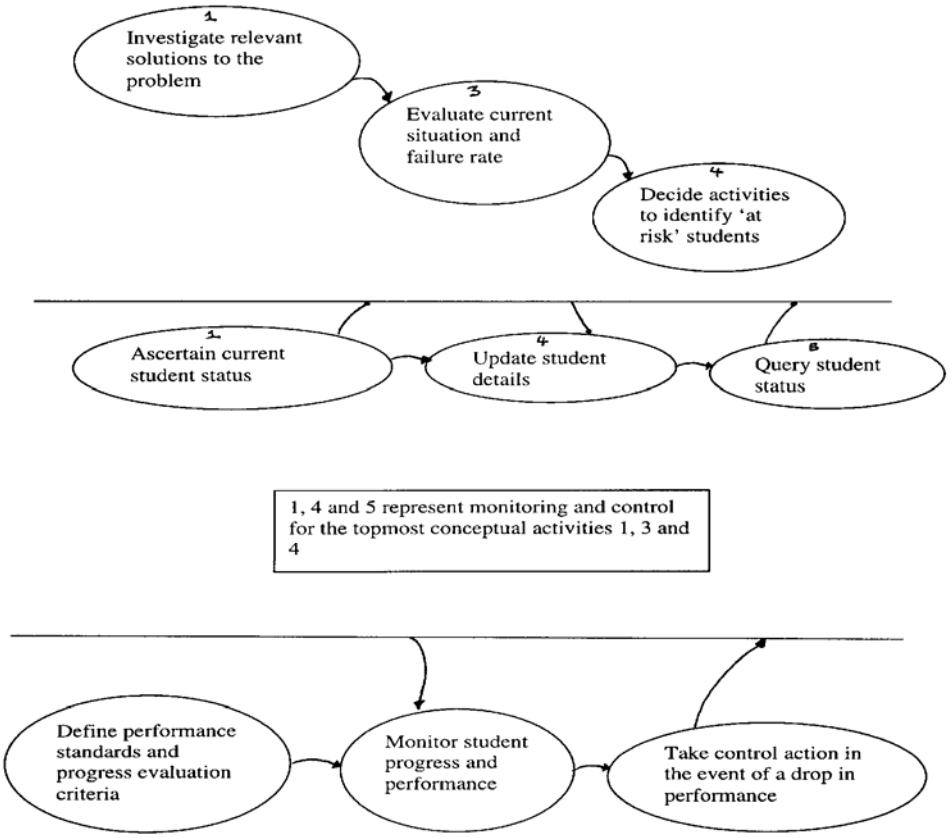


Figure 7.6: Revised nested conceptual model for the proposed system

This nested conceptual model is a unique occurrence. It is not one which the researcher has encountered in the existing SSM literature [refs]. This nested conceptual model depicts two (2) tiers of monitoring and control. It is literally two conceptual models. One is embedded within the other. The inner tier performs a dual role as both conceptual model and as a monitoring and control mechanism for the topmost conceptual model. The inner conceptual model then has its own monitoring and control mechanism. This would suggest that the duality of monitoring and control lends itself to a more efficient system in practice. This was not tested as the project went from analysis to design.

Measures of Performance

Effectiveness

Is this the right thing to be doing?

Does the monitoring and tracking improve the quality of dissertations and the number of successful projects?

Efficacy

Does the means work?

Does the monitoring and tracking work?

Efficiency

Is there minimum resource use?

There is minimum resource use because great advantages and benefits will be gained and there will be no need for expensive capital outlay. The resources needed are already present at the university and the tracking system will work in conjunction with the CAMS database. This is the University's Credit Accumulation and Management System (CAMS) within which all courses operate.

Comparison Phase

Conceptual	Reality
Ascertain current status of student	This status includes whether the student is fulltime or part-time among other things. Currently the postgrad. Admin has these details on the CAMS database. No other project management team member has electronic access. At the moment they pass hard copies of student details between each other
Update student details	The pathway leaders and project tutors have to pass this info to the postgraduate administrator who then does the update. They have no means to carry out this update themselves.
query student status	This query is done verbally by the project tutor or pathway leader and given to the postgraduate admin who then checks status. Sometimes registry or finance need to know a students status, if this is not on the system, the postgrad admin has to get this information from the project tutor or pathway leader.
Generate report of student status	this is done by the CAMS system used by the postgraduate admininistrator
Send periodical reminders of student status	No reminders of student status are sent at all. Any information is given at request of the relevant parties

Contact student for talk if necessary	This is done when students seem not to be doing well; but there is no prompting from the system when students have not handed in requested items. There are too many students for pathway leaders to effectively track students without help
monitor student's progress and performance	This is done by the supervisor, but not in a formal way
Define performance standard & progress evaluation criteria	Performance standard is already defined; but there is no formal progress evaluation criteria. This is left to the experience of the project management team
Take control action in the event of a drop in performance	This is done when supervisors recognise that students are not performing. They are written to and called in for discussion to see what can be done.

Table 7.7: Comparison phase for amplified Postgraduate Projects

Defining Changes

This proposed change is deemed 'systemically desirable' and 'culturally feasible' according to the criteria for defined changes stipulated by Checkland(1981) and Checkland and Scholes(1990). It has been derived from a comparison of the conceptual model with the reality of actual happenings in the School of Computing.

7.3 Proposed solution

A 'PostgradTrack' System that will enable the project management team to more effectively support the postgraduate project process. This system will enable them to know the current status of each postgraduate project student at any given point in time. This status could be defined in terms of various states. These could be active, suspended, terms of reference completed or pending. Fields could include also fulltime or part-time status, start and expected completion dates, supervisor, pathway leader, external client and a memo field for logging meetings with supervisors.

Reminders of students' status could be sent out periodically. Recommendations are that it could be every three months for part-time students and on a monthly basis for fulltime students. This would assist in the elimination of data redundancy and improve the accuracy and uniformity of source data. System rights would be given to the project tutor, Postgraduate Scheme Administrator, pathway leaders and academic supervisors. This should help to achieve clearer communication between administration and academic project staff.

7.4 Empirical Study Phase 2 – from Analysis to Design

This phase represents the design phase of the electronic system. Here the SSM outcomes are mapped to UML. The MoIST Project Option Selector Tool is again used to evaluate the project and select the most suitable project option.

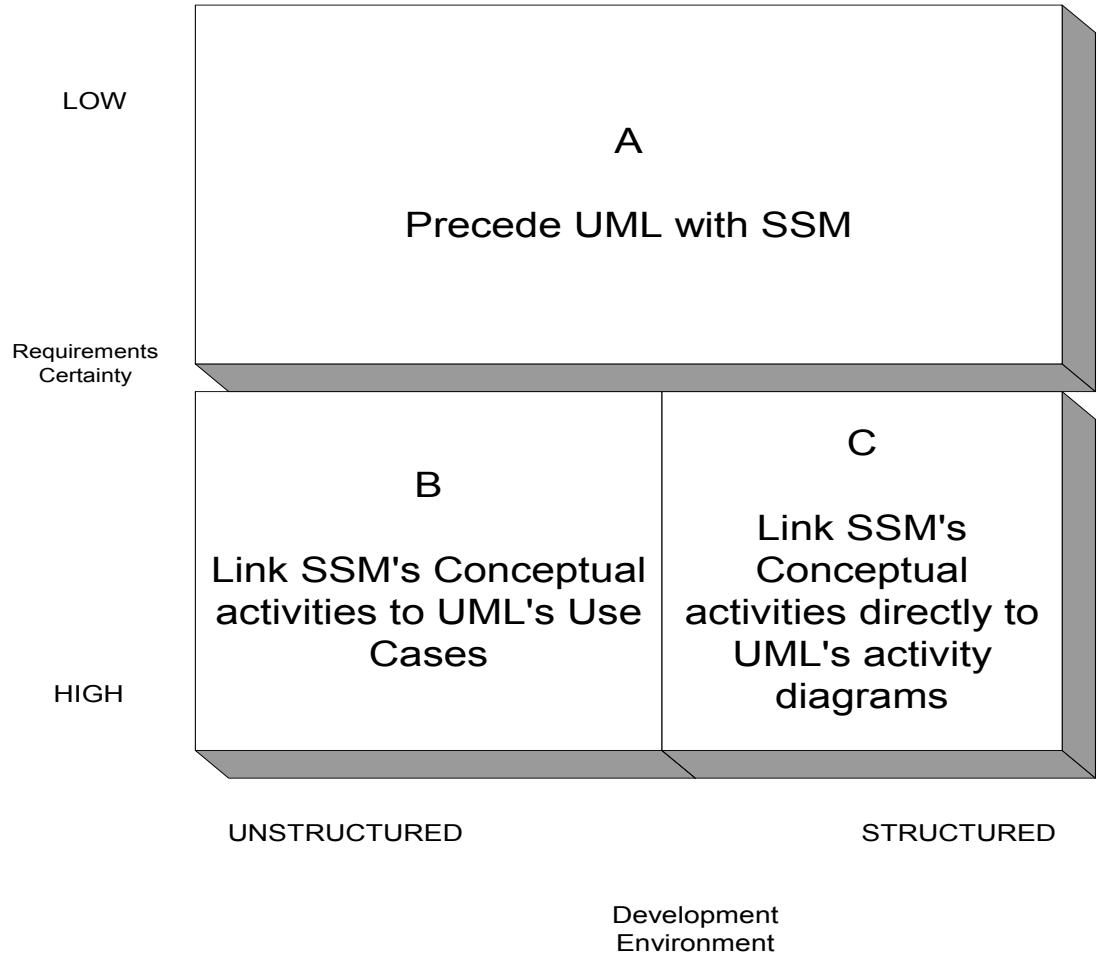


Figure 7.7: MoIST Method

7.7.3 Using MoIst's Project Option Selector Tool to get the best project option

- After the SSM findings, the characteristics of the Postgraduate Process Project were again analysed using the MoPros Selector Tool in order to make the most informed decision for this stage of the development process

MoIST's Project Option Selection Tool (MoPros)

	25 points	25 points	25 points	25 points	
Project Options	**Types of users**	**Developers' skillsets**	**Organizational environment**	**General characteristics**	**Total**
A	Users are a bit unsettled as they are experiencing organizational changes 0 points	Requirements at this point are not relatively clear to the development team 0 points	Development environment unstructured 15 points	Proposed system is to replace or enhance an existing system 10 points	25
B	Users uncertain about the need for the proposed system while others are more willing to be associated with it 20 points	Requirements known at this point are relatively clear to 80% of the development team. 18 points	Development environment has pockets of structure and unstructuredness. 20 points	Conflicting interests and the proposed system might cross functional borders 0 points	58
C	Users open to the new system 10 points	Requirements known at this information are very clear to 90% of the development team 0 points	Development environment is quite structured 10 points	Environment is relatively contention free 8 points	28

Table 7.5: MoIST's Project Option Selection Tool (MoPros)

Using the MoIST Project Option Selector Tool, it was found that the project requirements most closely matched project option B. **Option B** was chosen since Option A's activities were already completed within the SSM study conducted.

6.74 Option B: Enhance inception stage with SSM by deriving use cases from activities within the conceptual model.

Status: Requirements Certainty (High) + Development Environment (Unstructured)

MoIST Option B's Activities	
B1.	**Derive conceptual primary task model (CPTM).**
B2.	**Select and prioritise Conceptual model activities.**
B3.	**Determine which activities require further decomposition.**
B4.	**Determine which of the selected activities are candidates for IT support.**
B5.	**Identify actors.**
B6.	**Develop high-level use cases.**
B7.	**Develop multi-level use cases.**
B8.	**Identify high-level objects.**
B9.	**Map required high level services onto objects.**
B10.	**Continue design.**

7.7.5 Application of Option B within the MoIST method

6. Derive conceptual primary task model

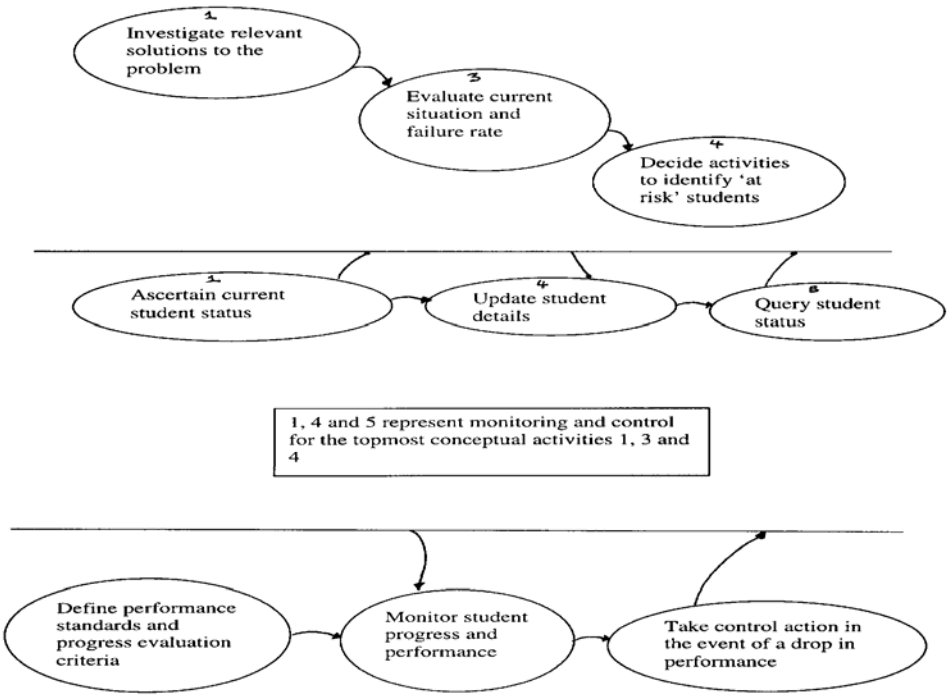
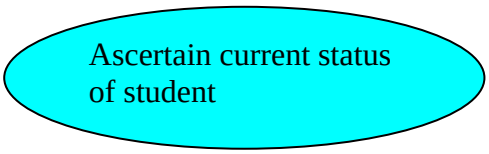


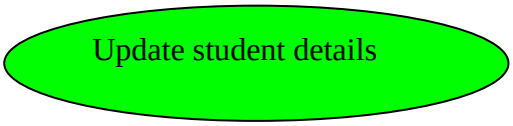
Figure 7.8: Conceptual model was derived from the SSM finding out stage.

7. Select and prioritise Conceptual model activities to be investigated for possible IT support

Priority 1



Priority 2



Priority 3

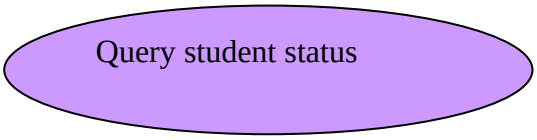


Figure 7.7: Priorities 1, 2 and 3

8. Identify scope or scale of OOA by determining which of the selected low-level activities are likely candidates for IT support. Also determine which may require further decomposition of some specific activities for which responsible use of IT is unclear

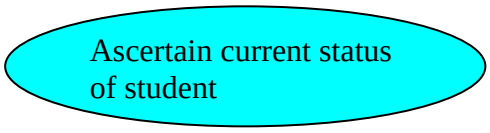
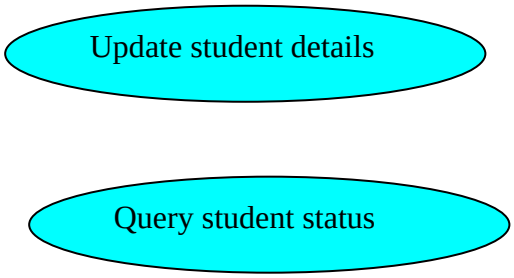


Figure 7.8: Activity in Option B of the MoIST Method



The selected low-level activities above are the most likely candidate activities for IT support

9. Identify actors for each of the low-level activities.

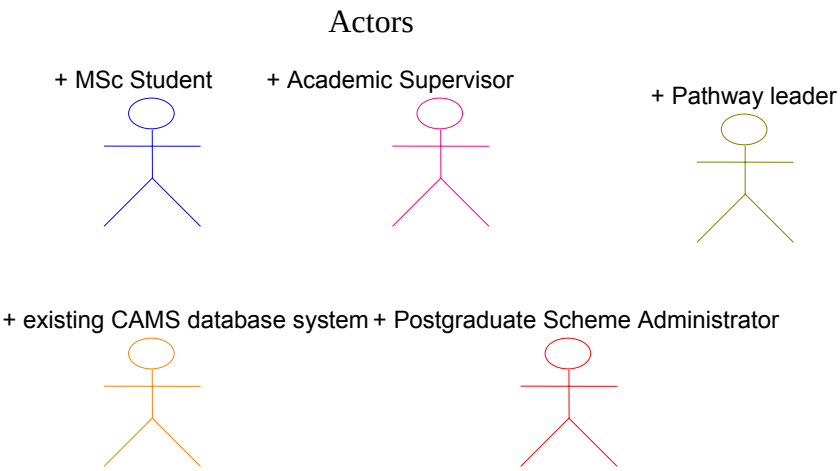


Figure 6.9: Actors for Low-Level Activities

Mapping SSM Human Activities to Use Cases for expanded conceptual model

- **Ascertain current status of student**
  - Search database for student based on an index for example surname
  - Click on appropriate search result
  - Retrieve appropriate status details

- **Update student details**
  - Retrieve relevant student details
  - Amend existing details

- **Query student status**
  - Perform query

- **Generate report of student's status**
  - Display report on screen
  - Select print option

- **Send periodical reminders of student status**
  - Acknowledge system alert
  - Email pathway leader and academic supervisor

- **Contact student for talk if necessary**
  - Email student

- **Monitor student's progress and performance**
  - Send email if deadlines are not met
  - Send email if student gets below 50%

- **Define performance standard and progress evaluation criteria**
  - Display the required grades and expectations on intranet
- **Take control action in the event of a drop in performance**
  - Email student for drop in talk

7.5 Conclusion

The MoIST method was shown to be workable in yet another context. The research project time did not allow for implementation of the system. The analysis and design work has been completed and documented. This can be used for future work on the system and will expedite development time.

The MoIST has again been shown to be effective in another real-world context. It can be safely concluded that the MoIST is an effective method.

Chapter 8 Conclusions and Future Research

8.1 Problem Introduction

The findings presented show that MoIST works in a different context. The next logical step is to summarize the findings and to point the direction for future research.

The major aim throughout the research has been to argue the necessity for human aspects to be reincorporated into hard systems engineering design. This is increasingly being neglected in most organizations involved in systems development (Mirijamdotter, 1998). This has resulted in innumerable software failures and tragedies that have impeded the success level of software development projects (Ewusi-Mensah, 2001).

8.2 Research Solution

The thesis argument started out by highlighting the fact that traditional hard systems engineering had a relatively good success rate. It then looked at how increasingly clients were not pleased by the resultant systems delivered to them. This delivery was more often than not eventually – say several years later – or not at all. This then led to the argument for a way of inculcating a systems thinking component within the traditional software development cycle. This would enable systems to be constructed more in accordance with the specification of the client. Chapters 2 to 6 of this thesis expounded on how the limitation of hard systems engineering was overcome by amalgamating SSM with the UML.

The combination of both the SSM and UML was looked at in chapters 4 and 5. It was merged into a new design method for integrating systems thinking with information systems design (MoIST). The MoIST method was applied in an academic organizational setting. This was at the University of Huddersfield, specifically the School of Computing and Engineering. The context was to examine the existing Academic Support Process in the University and ameliorate any problematic conditions discovered. The MoIST method was successfully applied and a major intervention was made which validated the action research carried out. This intervention made a radically improved difference in the existing system and was well received. An evaluation was then made of the MoIST's effectiveness in the situation. This was in terms of the electronic system's success, usability quotient and level of improvement to the former process. The conclusions gleaned from the research are summarised below.

8.3 Critical Appraisal of the Research

A critical appraisal of software development approaches and methods in general and this research in particular has been conducted. The findings indicate that though software technocrats and developers prefer the hard systems development approach, it has many problems. Firstly when used alone in the requirements elicitation phase, hard systems engineering models can encourage early design decisions before opportunities for improvement have been agreed. Also, SSM when used on its own in the requirements elicitation stage may lack some of the detailed information required by programmers. The most consistent and reliable feature of software development over the years have been its many failures. The appraisal covers select parts of the research including the MetricsMoIST Evaluator, Project MoIST's scales, activity diagrams, elaboration of use cases and lessons learnt from the two enactments of the MoIST method.

The MetricsMoIST Evaluator utilises Checkland's 5 E's Performance Indicator to assess the SSM artefacts produced by the MoIST. The Checkland's 5 E's may also be used to evaluate any resultant system produced from using MoIST.

Originally the MetricsMoIST Evaluator included metrics to evaluate both the SSM and UML components. However, objective feedback concerning the choice of metrics brought about the realisation that the UML metrics used, gave superficial or at best minimal evaluation. Using the lesson learnt from this, the UML metrics were then excluded leaving the Checkland's 5 E's component. The Checkland's 5 E's provide a loose guideline by which to measure the validity of the SSM artefacts produced. This will work more effectively for the method as the SSM component is still the unknown quantity for software development in general. This means that the SSM component is in need of more evaluation than the UML component for which a plethora of UML metrics currently exist. These include SDMetrics which is an object oriented design measurement tool for the UML and Fast&&Serious which is a UML based metric for effort estimation.

MoIST's scales were used in the MoIST's Project Option Selector Tool (MoPros) and the MetricsMoIST Evaluator using Chekland's 5 E's Performance Indicator. For the Selector tool (MoPros), a maximum of twenty-five (25) points are allocated to each of the four (4) sub-opions within each project option. For the MetricsMoIST evaluator, a maximum of twenty (20) points were allocated to each of the five (5) sub-

options. The weightings of twenty-five (25) and twenty (20) were chosen to help simplify and aid the scoring process by always yielding a percentage score. A percentage score was deemed to be easier for the project team to establish consensus on the individual scores of team-members. Expert feedback on the MoIST scales indicated that the scales could be rendered more appropriate. Consequently, though the MoIST project scales are the de facto recommended weightings, there is an alternative. The alternative recommendation is that project managers are free to use whatever weightings they deem to better facilitate a more successful scoring process. This applies to both the Selector Tool (MoPros) and MetricsMoist domains.

Two enactments of the MoIST method were undertaken for the research. The first was an exploration of an Academic Skills Process and the second was an exploration of a Postgraduate Project Process. Similar lessons were gleaned from both enactments of the method. One concerned the elaboration of use cases. The elaboration of use cases by relevant actors and domain experts was originally intended to be mandatory. In doing the research however, it proved to be an optional activity as the relevant actors were not engaged in the process of elaboration the use cases. This was done because the domain expert did not require help from the actors at that particular point in the intervention. The recommendation of the research is therefore for project managers or domain experts to evaluate each project on its own merit and decide whether or not there is a need to involve the relevant actors in the use case elaboration.

Another lesson learned concerned activity diagrams. In the UML-based artefact tool-kit that contains use-cases, sequence diagrams and activity diagrams, activity diagrams seem not to be as utilised as the other UML-based artefacts. This was seen in this research with the MoIST method. The two (2) case studies utilised Project Options A and B which had use-cases and sequence diagrams. Neither of the case studies tended towards Option C which utilised activity diagrams. This result suggests that for any possible future refinement or version of the MoIST method, Option C might be less relevant. Based on expert feedback, it could also potentially be omitted. In this research, it only served as a ‘control’ of sorts in the research experiment as Option C is more geared towards hard systems development where most of the requirements are known and the environment is structured. This might not always fit with most software development environments today as most software projects are unstructured in nature. Conversely, project options A and B are geared towards unstructured software development environments. This suggests a

plausible explanation concerning why both unstructured case studies utilised options A and B instead of C.

In the classification of the multi-methods in Chapter four, some methods were reviewed in terms of their proximity to the MoIST method. The methods deemed close to MoIST included the ISD Framework, RACE, Davis's Contingency framework, Multiview, RACE, BOOST and CCTA.

RACE needed to further tighten the linkage between the interaction models and formal models description language. MoIST rectified the RACE method linkage weakness by using a UML-based development environment to provide the linkage with SSM instead of LOTOS. This is significant as UML is more widely used commercially and more developers are familiar with it than LOTOS. The Davis's contingency framework examines several variables and determines which of its four options is best to use. The drawback with the framework it that it determines very well what option to use, but does not go on to say how the developer should follow the option to achieve the desired result. MoIST however rectifies this omission by detailing in a step by step manner exactly what to do to achieve each of the options chosen. The significance of this rectification is that it makes the software process clearer and makes it easier for the developers to follow the method. One of Multiview's drawbacks is that it uses ETHICS in its method structure. The significance here is that ETHICS though a popular method is no longer widely used in many commercial software environments. MoIST redresses this drawback by not using ETHICS in the soft to hard systems linkage, but by using the more current and ubiquitous UML-based linkage instead. One weakness of Lai's ISD framework is that it uses Martin-Odell's Object Oriented Analysis (OOA). This OOA however no longer provides the validity needed for current projects. This is because OOA has been subsumed into UML. MoIST rectifies this potential problem by utilising a UML-based context in which to link SSM. This is important as it potentially increases the chances of the method being utilised as UML is more widely used currently than OOA. BASE and BOOST only offer one core option for achieving the soft to hard linkage. This is significant as it does not necessarily provide software developers with the flexibility needed to maximize successful software development. MoIST rectifies this by providing more than one analysis and development options depending on the assessed characteristics of each software project. One of the problems with the CCTA approach is that SSADM is used for the 'hard systems' paradigm. This is significant as SSADM is no longer considered to be 'cutting edge' in its remit. MoIST

redresses this by using a UML-based context which is perceived as more relevant and current in the 'hard systems' domain of the software industry.

Hard systems engineering has from its inception played a vital role in the software industry. This by itself is insufficient to stem the rising tide of failures and incomplete systems. The MoIST method provides the requisite extension to make the software development process more effective. This is done in several ways. It allows for the combination of two strong approaches that promote a unified strength and subsumes each others deficiencies and limitations. It also alleviates criticisms levelled at the hard systems engineering approach. MoIST provided a firmer design structure for complex, unstructured situations. It additionally offers the potential for designing systems that will be more successful and pleasing to the clients. A major benefit gained from application of the MoIST method is an improved user definition for certain types of development project. This is an important prerequisite to successful implementation.

8.4 Further Work

In this research, SSM is shown to be a plausible basis for applying systems thinking and integrating it with a UML-based software development method. The MoIST method supports software developers and users in going from a complex, problematic organizational situation to the design of a new computer application suitably relevant to the situation. MoIST combines a set of viable options and methods into a coherent framework.

In summary, the new understanding of systems thinking integrated into information systems design suggests

- Complexity in any organisational or commercial setting can be reduced by utilising SSM and its activity modelling techniques.
- Design explanation can be used to provide the project manager and systems developer with the rationale behind the dynamics of the complexity in the requirements model.

The research findings are based on action research carried out using SSM. The current directions for future research are outlined as follows:

- **Consolidating theory** The MoIST method model will be linked to related studies in order to build sound theoretical bases for the model. Further investigation will be done of the MoIST model.

- **Further developing the new understanding** There is a need to study the dynamics of both essential and incidental complexity in relation to networks of different cognitive design activities described in the literature. This study will develop a sound theoretical foundation for our understanding of the requirements modelling process.

- **Testing the MoIST model and evaluating the new approach to using design explanation**. This model was identified from analysing qualitative data. However quantitative measurements are needed to confirm and strengthen the model. The model will be tested through quantitative empirical studies. A quantitative measurement of complexity to test the qualitative explanation will be conducted.

8.5 Research products or artefacts

- MoIST method
- ProcessMoIST
- MoIST Process Selector Tool
- ACcSys Electronic ILP System
- SSM artefacts including Conceptual models and overall results

8.6 Conclusion

The combination of SSM with a UML-based development method has already begun to produce academic research output and products that are making a solid difference in the world of software development.

The research produced two (2) comprehensive SSM studies. Some PhD theses that were read during the course of the research focused solely on the SSM study. (Kareborn,2002 and Mirijamdotter,1998). This research however went further and

produced a design method called MoIST. MoIST was used to link the results from one of its SSM studies to a UML-based method. Though the research was originally intended to end at the design of MoIST, it eventually proceeded to the subsequent design, implementation, deployment and testing of an electronic system called ACcSys.

The research is timely and fits in well with attempts by other researchers to address limitations of hard systems engineering approach and more specifically the UML.

Appendix

During the research several real-life SSM case studies were conducted. These were for learning purposes mainly and also with the hope that some would lead to major results. For the sake of space, not all could be included in the final written research document. The two SSM case studies expounded in chapters six and seven established that the MoIST method works. Both unstructured case studies demonstrated how options A and B worked in two separate real-life situations, but neither of them showed how option C works.

In order to demonstrate option C, this real-life SSM case study of the recruitment process below was used. It was really conducted according to Checkland's SSM guidelines up to conceptual model level. From that point on, the demonstration of how to go from conceptual modelling directly to activity diagramming is a 'made-up demo' for Option C. (this means that the last part was not done under research conditions).

A Study of the Recruitment Process in the School of Computing and Engineering

Introduction

Changing times and circumstances and political and social nuances have dictated the rise and the fall in the student admission levels at universities.

Over the last few years the levels seem to have fallen much more than they have risen. There is an intensive campaign on to attract more students to universities. Recruitment officers and admission offices are standard in most universities. International recruitment of students has become big business. Many universities have established marketing and publicity programmes overseas.

Background

The Department of Computing and the Department of Engineering were recently merged into one unit as the School of Computing and Engineering. A dean of school was appointed to head the unit. This incumbent was selected from the department of engineering. Much effort has been expended to make the school flow as a seamless unified whole. Over time though, the integration is appearing to not be as water tight as was originally intended.

An admissions office was formed to coordinate admissions efforts for the school. A school admissions officer was also appointed by the dean. This appointee formerly co-ordinated the admissions function in the Engineering department. The admissions officer position however has no mandate over the school admissions office as it is not a line management role. This means that the admissions officer cannot exert any influence over the daily running of the admissions office. It is probably significant that the admissions officer is in one physical location and the school admissions office in another. Investigation has shown that the two former departments had diametrically opposite modes of conducting the admissions process. We could call them 'the computing model of recruitment' vs 'the engineering model of recruitment'. The current admissions officer is still in favour of the engineering way of doing things while computing still maintains their admissions status quo. To this point there has been no official agreement between Computing and Engineering in this matter. It is quite telling that the admissions officer has initiated a name change of his function title from admissions coordinator to recruitment coordinator. He now sees them as two different functions. He sees his role as Recruitment Coordinator in the school as persuading students to come to the university in addition to increasing the recruitment levels in the school.

Methodology of the Research

The recruitment coordinator was interviewed to gain perspective on the recruitment process. Quite a comprehensive overview of how the process operates was given. This report will be followed by a subsequent one which will reflect the views of a wider cross-section of stakeholders of the recruitment process.

Soft Systems Methodology (SSM) is the methodology of choice in this study as it allows for exploration and understanding of the various issues involved in the school recruitment. It facilitates learning of the situation and enables a clearer view of the issues to be dealt with.

Soft Systems Methodology has been used with much success in many organizations to unravel complex is sues and bring structure to seemingly unstructured situations. It involves the stakeholders in the organization in the discovery process and enables them to own and proactively be the agents of change in their own organizations. Any changes that are to be made are usually more readily accepted and it has also been known to boost the morale of the human resource in an organization. SSM was made formulated and made popular by Professor Peter Checkland of Lancaster University.

Analysis of 2002 -2003 Academic year recruitment figures
(taken from report by Recruitment Coordinator)

Applications for computing courses in general have fallen. This is a national trend. This is currently happening probably as traditional universities widen participation to conform to the government's mandate and possibly because of bad positioning in the league tables owing to misleading statistics on completion rates
Medium to long term strategies need to be in place to attract students to apply in the first place. There seems to be little that admissions tutors can do in the short term to help with this
The 'local college' figures reflect a growing trend then it is clearly critical that students from the local catchment area of West Yorkshire are attracted and retained.
The mainstay courses still attract large numbers of students, though the reduction in applications is being felt in some of the IS and Computing courses.

June 2003 Applications numbers and conversion figures

	Applications	**Conversion rate**
New Media (including Games Programming)	592 (+235)	33.45
New Media (excluding Games Programming)	300 (-23)	33.33
Information Systems	516 (-110)	29.26
Computing	601 (+8)	29.28

this shows that there has been a fall in both the Information Systems and New Media(excl Games programming) applications.

Comparison of admissions afternoon visits with accepted offers

	Visits	Offers	Conversion
Information Systems & Computing	202	330	163%
New Media	228	198	87%

Analysis 1- Analysis of the intervention in the situation

Clients (Who caused the study to take place)	Dean of the School of Computing and Engineering, Recruitment Coordinator
Would-be problem solvers (who conducts the study)	Soft Systems Methodologist
Problem Owners (client + people with an interest in the situation	School of Computing and Engineering, recruitment coordinator, admission administrators

Analysis 2 - Social System Analysis

Social Roles	Behavioural Norms	Values that measure role performances as good or bad
Recruitment Coordinator	Organises all recruitment activities for the school and exists to persuade students to come to study at the university and to increase the flagging recruitment levels	Analyses past computing course admissions data in order to spot trends and be able to predict the future admissions patterns **GOOD** Able to effectively influence the way recruitment is carried out in the merged school of computing and engineering **GOOD** Does not have a comprehensive grasp of recruitment issues in the school **BAD**
Admissions Office Staff	Carry out all administrative duties for the school's admissions	Ensures that marketing and publicity documents are sent out well in advance **GOOD** Liaises well with relevant departments to help ensure top admissions rates **GOOD** Not in touch with what students are looking for in university schools and produce out dated and not trendy materials. **BAD**
Dean	Head of the school of Computing and Engineering	Has comprehensive knowledge of what goes on in the school and makes sound decisions based on that knowledge **GOOD** Able to lead the school in a manner that enables it to be

		competitive with other computing departments in other universities **GOOD** Out of touch with industry trends in computing and engineering **BAD**
Department heads	Oversee the running of all areas of their department	Coordinate all the staff under their jurisdiction efficiently **GOOD** Make unsound decisions that retard their department's growth **BAD**

Analysis three – examines the politics and power distribution in the organization

Disposition and Nature of Power

The main stakeholders in this process are the dean, the recruitment coordinator, the department heads and the admissions office staff. The dean wields the greatest authority here and has the clout to hire and fire and generally make any judgement call. The recruitment coordinator reports to the dean and was hired on the recommendation of the dean. The department heads report to the dean and the admissions office staff report to the department heads; not the former admissions coordinator. In effect therefore any changes that the recruitment coordinator sees fit to implement cannot be fully implemented without the agreement of the department heads. This could potentially lead to disagreement on how recruitment is carried out in the school as a whole. As it stands, it seems that the schools recruitment coordinator has a working jurisdiction over the engineering section of the school and none or not much over the computing section of the school. In the interest of the weltanschauung of the recruitment process, there needs to be agreement on a common recruitment modus operandi across the school. This will better enable the goal of maximising and increasing recruitment levels to be achieved.

C1- Derive Conceptual Primary Task Model

Formulating Root Definitions

Root Definition 1

"a system to standardise the way recruitment is carried out in the computing and engineering section of the school by establishing agreement on a common recruitment mode in order to make the recruitment process more effective and efficient"

Root Definition 2 - the chosen RD

"a system to improve the recruitment level in the School of Computing and Engineering by eliminating hindrances and bottlenecks to efficiency in recruitment in order to achieve highest recruitment levels possible"

CATWOE Analysis

C – Customer – Recruitment Coordinator, Dean

A – Actor - Recruitment Coordinator, Admissions Office Staff

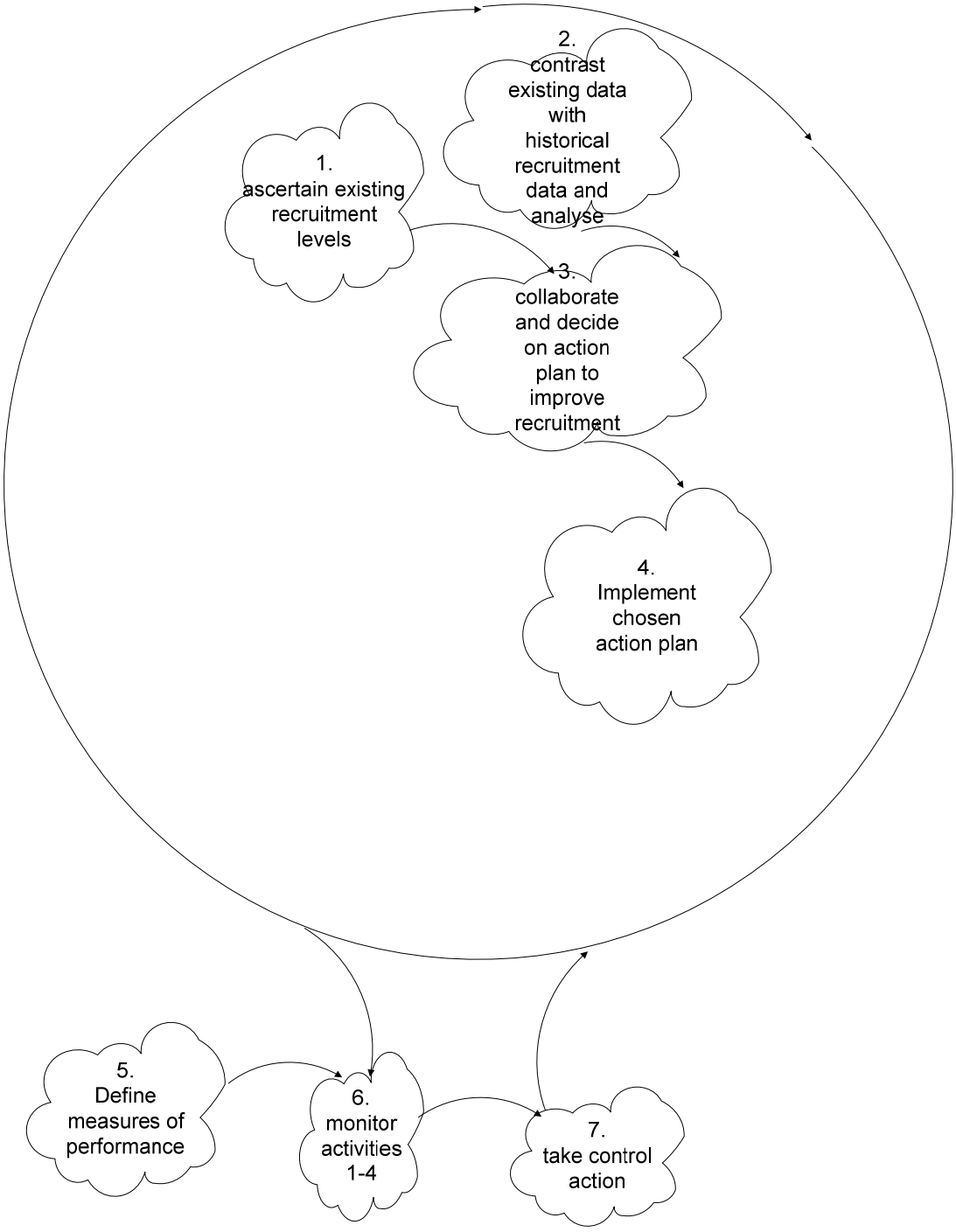
T – Transformation – Recruitment levels low --→ recruitment levels raised

W – Weltanschauung – acceptable recruitment levels are important for the successful future of the school and university

O – Owner – Dean

E – Environment – School and university recruitment policy

Conceptual Model



Comparison Phase

Conceptual	Reality
Ascertain existing recruitment levels	This is currently being done by the recruitment coordinator on a monthly basis
Contrast existing recruitment data with historical data and analyse	The recruitment coordinator performs this task and generates a monthly report by email
Collaborate and choose the most appropriate action plan to raise recruitment levels	There needs to be an increased level of collaboration in order to find the most effective way of improving recruitment
Implement action plan for recruitment	Actions to improve recruitment are currently being taken, but getting the benefit of a variety of competencies could drastically improve the process of recruitment

Defining Changes

Change 1

There is a need for improved efficiency in the way that recruitment levels are ascertained. This is in order to more quickly take advantage of the data and use it increase existing levels whilst cutting out inefficiencies.

Change 2

There could be increased collaboration among staff involved in the recruitment process. Such a collaboration would take advantage of each person's expertise and would result in a better way forward. Monthly meetings could be held with the department heads of the school, both computing and engineering to arrive at a common consensus and a more effective recruitment drive. Each course team could own the recruitment for their course. This would involve periodic meetings to see what is the best way to attract students. This would give the recruitment improvement drive the benefit of a wider cross- section of person's ideas.

C2 – Identify conceptual activities that are candidates for IT support

No	Selected conceptual activities
1	Ascertain existing recruitment levels
2	Contrast existing recruitment data with historical data and analyse
4	Implement action plan for recruitment

C2.2 – Look for conceptual activities that most closely match key areas gleaned from the study and the clients' needs

1	Ascertain existing recruitment levels

C3 – Identify nouns from conceptual activities

Existing, recruitment levels

C4 – link conceptual activities directly to the UML activity diagrams

No.	Conceptual activity statement	Activity state
1	The conceptual activity begins when the recruitment coordinator decides to ascertain existing recruitment levels by choosing the Continue function when the recruitment level details are displayed on the screen.	Display Current recruitment level; Get Recruitment request
2	The system requests that the recruitment coordinator enter the recruitment details, including: number of visits, number of offers, conversion figures, number of applications, conversion rates and any comments.	Display Recruitment Form
3	The recruitment coordinator chooses the visits and offers functions to arrive at the accurate recruitment details.	Get Recruitment Details
4	The system stores the recruitment details in the database.	Store New Recruitment Details
5	The system emails to the Recruitment Coordinator the recruitment details.	Email Recruitment Details.

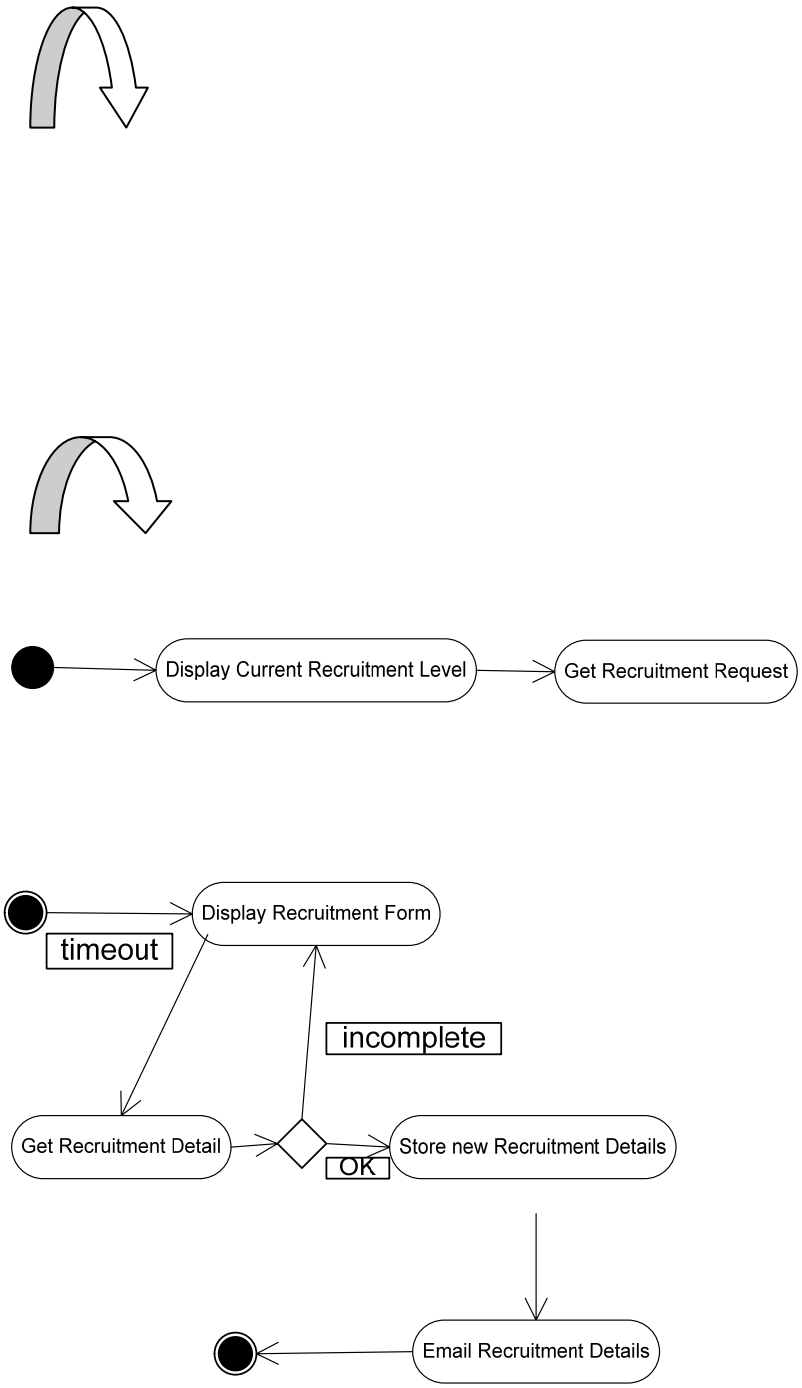


Fig 1 - Activity Diagram for Ascertaining Existing Recruitment Level

Display Recruitment Level is the initial activity state. The recursive transition on the state recognizes the fact that the display is continuously refreshed until the next transition fires (to *Get Recruitment Level Request*). This may be interpreted as the recognition of this state to be an activity, not an action. When in the state *Display Recruitment Form*, the timeout condition finishes the execution of the activity model. Alternatively, the state *Get Recruitment Details* is activated. If the recruitment details are incomplete, the system again enters the state *Display Recruitment Form*. Otherwise the system gets into the state *Store new Recruitment details* followed by the state *Email Recruitment Details* (the final state).

Bibliography

Ackoff, RL, 1998, A Systemic View of Transformational Leadership, Systemic Practice and Action Research, Vol. 11, No. 1, 1998

Abdel-Hamid, T. K., and S. E. Madnick. 1990. The Elusive Silver Lining: How We Fail to Learn from Software Development Failures. *Sloan Management Review* 32(1): 39-48.

Ambler, S.W. (2002) *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. John Wiley & Sons

Ambler, S, 2001, The Object Primer – The Application Developer's Guide to Object Orientation and the UML, 2nd Ed, Cambridge University Press

Archer, R and Bowker, P, 1995, BPR Consulting: an evaluation of the methods employed, Business Process Re-engineering & Management Journal, Vol. 1, No. 2, pp. 28-46

Armarego, J, 1999, Educating Requirements Engineers in Australia: A Critical Study, http://eng.murdoch.edu.au/~jocelyn/papers/phd_proposal.doc

Armarego, J and Clarke, S, 2002, Preparing Students for the future: learning creative software development – setting the stage http://eng.murdoch.edu.au/~jocelyn/papers/armarego.herdsa.pdf

Avison, D.E., Baskerville, R. and Myers, M.D, 2001 "Controlling action research projects," *Information Technology & People* (14:1), 2001, pp. 28-45

Avison, D and Fitzgerald, G, 1995, Information Ssytems Development: Methodologies, techniques and Tools, (2nd ed) McGraw-Hill, London

Avison, D.E. and Wood-Harper A.T. (1990) *Multiview: An exploration in Information Systems Development*., McGraw-Hill, Maidenhead.

Barton, J, Emery, M, Flood,RL, Selsky, JW and Wolstenholme, E, 2004, A Maturing of Systems thinking? Evidence from three perspectives, Systemic Practice and Action Research, Vol.17, No.1, February 2004

Baskerville, R., and Myers, M.D.2004, "Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice-Foreword," *MIS Quarterly* (28:3) 2004, pp 329-335.

Baskerville, R. and Pries-Heje, J. 1999, "Grounded action research: a method for understanding IT in practice," *Accounting, Management and Information Technologies* (9:1), 1999, pp. 1-23.

Baskerville, R.L. and Wood-Harper, A.T. 1998, "Diversity in information systems action research methods," *European Journal of Information Systems* (7), 1998, pp. 90-107.

Baskerville, R.L. and Wood-Harper, A.T.1996, "A Critical Perspective on Action Research as a Method for Information Systems Research," *Journal of Information Technology* (11), 1996, pp. 235-246.

Batra, D and Marakas, GM.,1995, Conceptual data modelling in theory and practice. European Journal of Information Systems. 4, 185-193

BBC News, July 8, 2004
news.bbc.co.uk/1/hi/uk/3876507.stm

Beck, K., *Extreme Programming Explained – Embrace Change*. Reading, M.A.: Addison-Wesley. 2000

Bell, S and Wood-Harper, T (2003), How To setup Information Systems: A non-specialist's guide to the Multiview Approach, Earthscan Publications Ltd.

Bell, S., and Wood-Harper, A.T., 1992, Rapid Information Systems Development, McGraw-Hill, Maidenhead, 1992.

Bennett, S., McRobb, S. and Farmer, R., Object-Oriented Systems Analysis and Design, (2nd Ed.), McGraw-Hill, Maidenhead, 2002.

Bennett, S, Skelton and Lunn, K, 2001, Schaum's Outline of UML, McGraw-Hill

Bennett, P, Ackermann, F, Eden,C and Williams, T, 1997, Analysing Litigation and Negotiation: Using a Combined Methodology , Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Bergvall-Kareborn, B, Mirijamdotter, A and Basden, A, 2004, Basic Principles of SSM Modelling: An Examination of CATWOE from a Soft Perspective, Systemic Practice and Action Research, Vol. 17, No. 2, April, 2004

Bergvall-Kareborn, B, 2002, Qualifying Function in SSM Modelling – A Case Study, Systemic Practice and Action Research, Vol. 15, No. 4, August 2002

Bergvall-Kareborn, B., 2002, *A Multi-Modal Approach to Soft Systems Methodology*, Doctoral dissertation, Department of Business Administration and Social Science, Lulea University of Technology, 2002.

Boggs, W and Boggs, M, Mastering UML with Rational Rose, Sybex, 2002(electronic version)

Booch, G., 1998, Best of Booch – Designing Strategies for Object Technology, Cambridge University Press, 1998.

Booch, G., 1994, *Object Oriented Analysis and Design with Applications* (2nd Ed.), Menlo Park, CA: Benjamin/Cummings, 1994.

Booch, G, Rumbaugh, J and Jacobson, I, 1998, The Unified Modelling Language User Guide, Addison-Wesley Professional

Bouzeghoub, M, Gardarin, G,Valduriez, 1997, Object Technology:Concepts and Methods, International Thomson Computer Press

Bowl, M, 2003, Non-traditional Entrants to Higher Education: They Talk About People Like Me, Trentham Books.

Brocklesby, J,1997, Becoming Multimethodology Literate: An Assessment of the Cognitive difficulties of working across paradigms, Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Brooks, Frederick, Jnr, 1995, The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition, Addison-Wesley Professional.

Brown-Syed, C., 1993, *Soft, Appreciative, and General Systems: Idealism in Action*, Third Canadian Conference on Foundations and Applications of General Science Theory, Ryerson Polytechnic University, Toronto, Ontario, Canada, 1993

Bruegge, B & Dutoit, 2000, A, Object-Oriented Software Engineering – Conquering Complex and Changing Systems, Prentice Hall, New Jersey, 2000.

Bryant, A, 1996, Introduction and Overview -The Projects of Methods Integration, Proceedings of the Methods Integration Workshop, Leeds, 25-26 March, Bryant, A & Semmens (Eds), Electronic Workshops in Computing, Springer

Bubenko, J. (1995). *Challenges in Requirements Engineering: keynote address.* Paper presented at the RE'95:
Second IEEE International Symposium on Requirements Engineering, York (UK).

Bustard, DW, Holcombe, M and Sommerville, I, 2004, BoF: New Directions in UK Software Engineering Research, Proceedings of the 26th International Conference on Software Engineering (ICSE'04), IEEE

Bustard, DW, He, Z and Wilkie, FG, Linking Soft Systems and Use-case Modelling through Scenarios, Interacting with Computers, Vol. 13, No.1, Elsevier, October 2000, pp. 97-110

Bustard D.W., Kawalek, P and Norris, M.T. (Eds.), 2000, Systems Modelling for Business Process Improvement, Artech House, May 2000.

Bustard, DW, He, Z and Wilkie, FG, 1999, Soft Systems and Use-Case Modelling: Mutually Supportive or Mutually Exclusive?, Proceedings of the 32nd Hawaii International Conference on System Sciences (HICSS-32), Maui, Hawai (CDROM), IEEE, January 1999, 8 pages.

Bustard, D. W., Dobbin, T. J., and Carey, B. N., 1996, "Integrating Soft Systems and Object-Oriented Analysis", *IEEE International Conference on Requirements Engineering*, Colorado Springs, Colorado, April, 1996, pp. 52-59

Bustard, D. W. and Lundy, P.J., 1996, Integrating Process Modelling and Soft Systems Analysis, Methods Integration Workshop, Leeds, March 1996

Bustard, D, 1994, Progress towards RACE: A 'Soft-centred' Requirements Definition Method, Software Quality and Productivity, 1994, pages 29-36.

Buzan, T., Buzan, B. (2000), *The Mind Map Book, Millennium edition*, London: BBC Worldwide

Cantor, M., 1998, Object-Oriented Project Management with UML, John Wiley & Sons, Canada, 1998.

Cao, G, Clarke,S and Lehaney, B, 2004, The Need for a Systemic approach to Change Management - A Case Study, Systemic Practice and Action Research, Vol 17, No.2, April 2004

Carroll, J M and Swatman, P A (1999*) Opportunism in the Requirements Engineering process* School of Management Information Systems Working Paper 1999/02, Deakin University, Australia

Carroll, J., (Ed.), 1995, *Scenario-Based Design: Envisioning Work and Technology in System Development,* New York: John Wiley, 1995.

CCTA, 1993, *Applying Soft Systems Methodology to an SSADM Feasibility Study*. HMSO, Crown Copyright, 1993. London

Champion, D and Stowell, FA, 2003, Validating Action Research Field Studies: PEArL, Systemic Practice and Action Research, Vol 16. No.1, February 2003

Champion, D and Stowell, F, 2002, Navigating the Gap Between Action and a Serving Information System, Information Systems Frontiers 4:3, 273-284, 2002, Kluwer Academic Publishers

Champion, D and Stowell, FA, 2001, PEArL: a systems approach to demonstrating authenticity in information systems design, Journal of Information Technology, Routledge, Vol 16, Number 1, March, 1, 2001, pp 3-12

Checkland, P, 2000, The Emergent Properties of SSM in Use: A Symposium by Reflective Practitioners, Systemic Practice and Action Research, Vol. 13, No. 6, 2000

Checkland, P interviewed by Mark Winter, Human Resource Development International, 3(3), pp. 411-417

Checkland, PB and Scholes, J, 1999, Soft Systems Methodology in Action with a Thirty Years Retrospective on SSM, John Wiley, Chichester

Checkland, P.,1997, *unpublished presentation given at Systems for sustainability: People, Organisations and Environments,* 5th International Conference of the United Kingdom Systems Society, Milton Keynes: The Open University, July 1997.

Checkland, PB and Holwell, S, 1998, Information, Systems and Information Systems: Making Sense of the field, John Wiley & Sons, Chichester

Checkland, P and Holwell, S, 1998, Action Research: its nature and validity, Systemic Practice and Action Research, 11(1), pp. 9-21

Checkland, PB.,1981, *Systems Thinking, Systems Practice*, John Wiley, Chichester

Checkland, PB and Holwell, S, 1993, Information management and organizational processes: an approach through soft systems methodology, Journal of Information Systems, 3, 1-15

Checkland, PB., and Scholes, J, 1990, *Soft Systems Methodology in Action*, John Wiley, Chichester

Checkland, P., 1991 "From framework through experience to learning: the essential nature of action research," in *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H-E. Nissen, H.K. Klein, R.A. Hirschheim (eds.), North-Holland, Amsterdam, 1991, pp. 397-403.

Checkland PB, 1988, Information systems and systems thinking: time to unite? International Journal of Information Management, 8: 239-248

Checkland, P and Casar, A, 1986. Vickers' Concept of an Appreciative System: A Systemic Account. Journal of Applied Systems Analysis 13:3-17

Chesney, T and Fletcher, H, 2000, Process Differentiation and Information Systems Development, Proceedings of the 33rd Hawaii International Conference on System Sciences.

Coad, P and Yourdon, E, 1990, Object Oriented Analysis, Yourdon, Englewood Cliffs, NJ.

Cockburn, A. ,2002, *Agile Software Development*. Addison Wesley Professional

Cockburn A., 2001,*Writing Effective use cases*. Addison Wesley, 2001.

Cockburn, A,1999, Characterizing People as Non-Linear, First-Order Components in Software Development(Technical Report to be submitted for external publication)

Cockburn A., 1997, Structuring use cases with Goals. *Journal of Object Oriented Programming.* Sep-Oct and Nov – Dec. SIGS Publications, 1997.

Computer Weekly, June 8, 2004
www.computerweekly.com/Article131375.htm

Cooke, J, 1996, Methods Integration: Time for Reflection (and Reorientation?), Proceedings of the Methods Integration Workshop, Leeds, 25-26 March, Bryant, A & Semmens (Eds), Electronic Workshops in Computing, Springer

Cropley, D, Yue, Y and Cook, S, 2003 On identifying a Methodology for Land C2 Architecture Development, Land Warfare Conference, Adelaide, October 2003.

Cybulski, J. L., Nguyen, L., Thanasankit, T., and Lichtenstein, S., 2003, Understanding problem solving in requirements engineering, Burwood, Australia: School of Information Systems, Deakin University.

Damian, D, Zowghi, D, Vaidyanathasamy, L and Pal, Y, 2004, An Industrial Case study of immediate benefits of Requirements Engineering Process improvement at the Australian Centre for Unisys Software, Empirical Software Engineering, 9, 45-75, Kluwer Academic Publishers, Netherlands

Davis, GB, 1982, Strategies for Information Requirements Determination, IBM Systems Journal, Vol. 21, No. 2, pp. 4-30

Dawson, C, 2000, The Essence of Computing Projects: a student's guide, Prentice Hall

De Bono, E., 2000, Six Thinking Hats, Penguin.

DFes, 2003, The Future of Higher Education, White paper. http://www.dfes.gov.uk/hegateway/strategy/hestrategy/pdfs/DfES-HigherEducation.pdf

Dick, B. and Swepson, P., Appropriate validity and its attainment within action research:an illustration using soft systems methodology [online].1994. Available at http://www.scu.edu.au/schools/gcm/ar/arp/sofsys2.html

Dobbin, TJ and Bustard, DW, 1994, Combining Soft Ssytems Methodology and Object-Oriented Analysis: The search for a Good fit, Proceedings of the 2nd Information Systems Methodologies Conference, Edinburgh, August 1994

Dobson, J and Strens, R,1994, Organisational requirements definition for information technology systems, in Proceedings of the first International Conference on Requirements Engineering, 18-22 April, Colorado Springs, Colorado, IEEE Computer Society Press, Los Alamitos CA, 158 -65

Doyle, K.G., Wood, J.R.G. and Wood-Harper, A.T. Soft systems and systems engineering: on the use of conceptual models in information system development. J of Info Systems, 1993 3, 187-198

Doyle, K and Wood, RJ, 1991, Systems thinking, systems practice, dangerous liaisons. Systemist, Vol. 13, No.1, pp. 28-30

Dittrich, Y, Floyd, C and Klischewski, R(Eds): Social Thinking – Software Practice. MIT Press, 2002

Downs, D and Lunn, K., 2002, Analysis and Design for Process Support Systems using Goal-oriented Business Process Modelling, *Workshop on Goal-Oriented Business Process Modeling (GBPM'02),* Toronto (Canada), May 27- 28, 2002,

D'Souza, DF and Wills, AC, 1998, Objects, Components and Frameworks with UML: The Catalysis Approach, Addison Wesley, Harlow

Drummond, H., 1996, The Politics of Risk: Trials and Tribulations of the Taurus Project. *Journal of Information Technology* 11: 347-357.

Eliens, A, 2000, Principles of Object-Oriented Software Development, 2nd Ed., Addison-Wesley

Ewusi-Mensah, K, 1997, Critical issues in abandoned information systems development projects. Communications of the ACM, 40(7): 74-80.

Ewusi-Mensah, K., 2003, Software Development Project Failures: Anatomy of abandoned projects, MIT Press

Ewusi-Mensah, K, 1997, Critical issues in abandoned information systems development projects. Communications of the ACM, 40(7): 74-80

Ferrari, FM, Fares, CB and Martinelli, DP, 2002, The Systemic Approach of SSM: The Case of a Brazilian Company, Systemic Practice and Action Research, Vol 15, No. 1, February 2002

Flick U., 1999, *An Introduction to Qualitative Research*, SAGE Publications, London, 1999.

Flood, R.L.,(2000). A Brief Review of Peter B. Checkland's Contribution to Systemic Thinking. Syst. Pract. 13(6), 723-731

Flood, RL, 1999, Knowing of the Unknowable, Systemic practice and Action Research, Vol. 12, No. 3, 1999

Flood, R and Romm, N, 1997, From Metatheory to Multimethodology, Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Flood, RL and Jackson, MC, 1991, Creative Problem Solving: Total Systems Intervention.

Floyd, C, 1987, Outline of a paradigm change in software engineering. 191-210 in G Bjerknes et al. (Eds.) Computers and Democracy. Avebury:Aldershot.

Flynn, D., 1998, Information Systems Requirements: Determination and Analysis, (2nd Ed.), Maidenhead:McGraw-Hill, 1998.

Fowler, M., 2003, UML Distilled: A Brief Guide to the Standard Object Modelling Language (3rd Ed.), Addison-Wesley Professional

Fowler, M. and Scott, K., 2000, *UML Distilled(2nd Ed.)*. Reading, M.A.: Addison-Wesley. 2000

Fowler, M. and Scott, K, 2000, UML Distilled: A Brief Guide to the Standard Object Modelling Language (2nd Ed.), Addison-Wesley Professional.

Fuenmayor, R, 2000, A Brief Crack of Light? Systemic Practice and Action Research (13) 6

Galliers, R D, 1997. Against Obliteration – Reducing risk in business process change. In steps to the Future – Fresh thinking on the management of IT-based organisational transformation, edited by Sauer, C., Yetton, P. w. AND Associates, a. San Francisco:Jossey-Bass Publisher.

Galliers, RD, 1992, Soft Systems, Scenarios, and the Planning and Development of Information Systems, Systemist, Vol 14, No. 3, pp. 146-59

Galliers, R.D and Land, F.F, 1987 Choosing Appropriate Information Systems Research Methodologies, Communications of the ACM. 30(11), 900-902.

Geddes & Grossett, English Dictionary, 1999.

Gharajedaghi, J., 1999, *Systems Thinking: Managing Chaos and Complexity: A platform for Designing Business Architecture*.Butterworth-Heinemann,1999.
Gill, A, 1997, Managing a Virtual Organization, Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Glass, R. L., 1995, A theory about software's practice (Editor's Corner). *Journal of Systems and Software, 28*, 187-188.

Gold, J, 2001, Storying Systems: Managing Everyday Flux Using Mode 2 Soft Systems Methodology, Systemic Practice and Action Research, Vol 14, No. 5, October 2001

Graham, I, 1998, Requirements Engineering and Rapid Development, Addison Wesley, London

Hammer, M and Champy, J. 1995. Reengineering the corporation: a manifesto for business revolution. Rev. ed. London: Brealey.

Hansson, T, 2003, Learning by Action Research: A Policy for School development, Systemic Practice and Action Research, Vol. 16, No.1, February 2003

Hawryszkiewycz, I, 2001, Introduction to Systems Analysis and Design, 5th Ed., Prentice Hall

Henderson-Sellers, B and Unhelkar, B, 2000, OPEN modelling with UML, ACM Press, Addison-Wesley

Hindle, T., Checkland, P., Mumford, M. and Worthington, D. "Developing a Methodology for Multidisciplinary Action Research: A Case Study," *Journal of the Operational Research Society* (46:4), 1995, pp. 453-464

Hirscheim, R, Klein, HK and Lyytinen, K, 1995, Information Systems Development and Data Modelling:Conceptual and Philosophical Foundations, Cambridge University Press, Cambridge

Holst, M, Mirijamdotter, A, Bergvall-Kareborn, B, Oskarsson, H. 2004. Information and Communication Technology in Dynamic Organisations., IRIS27, 2004

Holwell, S, 2000. Soft Systems Methodology:Other Voices. Syst Pract. 13(6), 773-797

Holwell, S,1997, Soft Systems Methodology and its role in Information Systems. Doctoral Thesis, Lancaster University, Lancaster.

Hopkins, J, 1999, Does the Aptitude Test offered by the School of Computing predict Students' Performance in that Programme? Postgraduate Diploma in Education (PGDip) thesis, University of Technology, Jamaica

Hopkins, J., *Providing a semantic model of the Unified Modelling Language 1.3 by establishing a semantic description of the existing syntactic standard*, M.Sc. dissertation, 2001.

Hopkins, J and Wade, S, 2004, A Study of the Postgraduate Project Process in a School of Computing. Proceedings of the 7th IASTED International Conference on Computers and Advanced Technology in Education. August 16-18, Kauai, Hawaii, USA. (presented)

Hopkins, J and Wade, S, A Successful Intervention in the Academic Learning Support Process in a School of Computing through the utilisation of Soft Systems Methodology (SSM), Journal of Advanced Technology, Vol., No., pp:, September 30, 2004, IADAT Publishers. ISSN 1698-1073

Hopkins, J and Wade, S, Major Improvements to the Academic Learning support process in a School of Computing through the utilisation of Soft Systems Methodology (SSM), Proceedings of the International Association for the Development of Advancement in Technology conference on Education (IADAT-e2004), July 7 – 9, 2004, Bilbao, Spain (presented)

Hult, M. and Lennung, S.A., 1980, Towards a Definition of Action Research: A Note and Bibliography, *Journal of Management Studies*, May 1980, pp. 241-250

Jackson, MC, 2003, Systems Thinking: Creative Holism for managers, Wiley Europe

Jackson, MC, 2000, Notes and Insights – Checkland, Peter Bernard (1930-), Systems Research and Behavioural Science, Syst. Res. 17, S3-S10 (2000)

Jackson, MC, 2000, Systems Approaches to Management, Kluwer/Plenum, New York.

Jackson, M C, 1997, Pluralism in Systems Thinking and Practice, Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Jackson, MC, 1991, Systems Methodology for the Management Sciences, Plenum, New York, NY.

Jackson, M.C., 1991, The Origins and Nature of Critical Systems Thinking. Systems Practice. 4, 131-149.,.

Jackson, MC, Flood, RL, Mansell, GJ, and Probert, SVE (Eds.), 1991, Systems Thinking in Europe, Plenum.

Jacobson, I., 2000, The Road to the Unified Software Development Process, Cambridge University Press & SIGS Books, 2000.

Jacobson, I., Booch, G. and Rumbaugh, J., 1999, *The Unified Software Development Process*. Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G.,1992, *Object-oriented Software Engineering: a use case driven approach*, Addison-Wesley, Reading, Mass.

Jacobson, I, 1995. "Use Cases in Large Scale Systems", *Report on Object Analysis and Design,* 1995 1(6), pp. 9-12.

Jacobson, I, 1995, The Object Advantage

Jayaratna, N., *Understanding and Evaluating Methodologies; NIMSAD: A Systemic Framework,* Maidenhead:McGraw-Hill, 1994.

Keys, P. and Roberts, M., Information Systems Development and Soft Systems Thinking: towards and improved methodology. In *Systems Thinking in Europe,* Plenum, London. 1991.

Kock, NF, Jr., McQueen, RJ and Scott, JL, 1997, Can Action Research be made more rigorous in a positivist sense? The contribution of an Iterative Approach, Journal of Systems and information Technology, V.1, No. 1, pp. 1-24

Kom, J, 1999, Qualitative modelling of information systems. In Synergy Matters: Proceedings of the Sixth International Conference of the UKSS, Castell, AM, Gregory, AJ, Kindle, GA, James, ME and Ragsdell, G (eds), Plenum, New York, pp. 571-6.

Kotonya, G. and Sommerville, I., 1998 *Requirements Engineering: processes and techniques*, John Wiley.

Krutchen, P., 2000. *The Rational Unified Process – An Introduction*. 2nd Edition. Addison-Wesley.

Lane, C and Galvin, K, 1999 Methods for Transitioning from Soft Systems Methodology (SSM) Models to Object Oriented Analysis (OOA), developed to support the Army Operational Architecture (AOA) and an Example of its Application, Command and Control Research and Technology Symposium, U.S. Naval War College, Rhode Island – June 29 – July 1, 1999.

Lang, N, 1993. Shlaer-Mellor Object Oriented Analysis Rules, A Sigsoft Software Engineering notes Vol 18 no. 1, Jan 1993.

Larman, C. *Applying UML and Patterns. An Introduction to Object Oriented Analysis and Design and the Unified Process*. Prentice-Hall PTR. 2nd Edition 2001

Larsson, NO and Malmsjo, A, 1998, A Model for Design of Human Activity Systems, Systemic Practice and Action Research, Vol. 11. No 4, 1998

Leon, A., 2000, SDLC – A Guide to Software Configuration Management, Artech House

Lewis, P, 1994, Information Systems Development, Pitman, London.

Lewis, P.,1993, Linking of Soft Systems Methodology with data-focussed information systems development. *Journal of Information Systems* 3 (3) 169-86. 1993;

Liang, Y, West, D and Stowell, FA, 1998, An interpretivist approcach to IS definition using object modelling, Information Systems Journal, 8, 163-80

Lorenz, M. and Kidd, J., 1994, Object-Oriented Software Metrics: A Practical Guide, Englewood Cliffs, NJ: Prentice-Hall, 1994.

Lunn, K, 2003, Software Development with UML, Palgrave, MacMillan

Maciaszek, L, 2001, Requirements Analysis and Systems Design: Developing Information Systems with UML, Addison Wesley

Mann, J., 1996. *The Role of Project Escalation in Explaining Runaway Information Systems Development Projects: A Field Study.* Georgia State University.

Mansell, G, 1991, Action Research in information systems development, Journal of Information Systems (1), 1991, pp. 29-40

Mathiassen, L, 2002, Collaborative Practice Research, Information Technology and People (14:4), 2002, pp. 321-345.

Mathiassen, L, Munk-Madsen, A, Nielsen, P and Stage, J, 2000, Object-Oriented Analysis and design, Marko Publishing ApS, Aalborg.

Mathiassen, L and Nielsen, PA, 2000, Interaction and transformation in Soft Systems methodology, Systems Research and Behavioural Science, Vol. 17, pp. 243-53

Mathiassen, L, Munk-Madsen, A, Nielsen, P.A and Stage, J, 1994, Combining two approaches to Object Oriented Analysis, Proceedings of the International Symposium on Object-Oriented Methodologies and Systems, September 21-22, 1994, pages 158-170

May, S and Bousted, M, 2004. Investigation of Student Retention through an Analysis of the First Year Experience of Students at Kingston University: Widening Participation and Lifelong Learning, The Journal of the Institute for Access Studies and The European Access Network, Vol 6, No 2. ISSN 1466-6529, August 2004

Maxwell, J 2003, Thinking For a Change: 11 Ways Highly successful people approach life and work, Warner Books Inc, NY

Melao, M. and Pidd, M., 2000 *A conceptual framework for understanding business processes and business process modelling*, Information Systems Journal 10, 105-129, 2000.

Midgley, G, 2003, Science as Systemic Intervention: Some Implications of Systems Thinking and Complexity for the Philosophy of Science, Systemic Practice and Action Research, Vol 16, No.2, April 2003

Miles, R, 1992, Combining hard and soft systems practice:grafting and embedding revisited, Systemist, Vol. 14, No. 2, pp. 62-6

Miles, R., 1988, Combining “hard” and “soft” systems practice: grafting or embedding? *Journal of Applied Systems Analysis* 15 pp 55-60.

Mingers, J. 2001, "Combining IS Research Methods: Towards a Pluralist Methodology," *Information Systems Research* (12:3), 2001, pp. 240-259

Mingers, J., 2000, An idea ahead of its Time: The History and Development of Soft Systems Methodology. *Systemic Practice and Action Research*, Vol 13, No. 6, 2000;

Mingers, J, 1997, Multi-paradigm Multimethodology. MultiMethodology. Mingers J. and Gill, A., Eds. Chichester, John Wiley & Sons:1-20

Mingers, J., and Brocklesby, J, 1996,. Multimethodology: Towards A Framework For Critical Pluralism. *Systemist, Vol.18, Number 3, August 1996,101-131*.

Mingers, J, 1995, Using soft systems methodology in the design of information systems. In Information Suystems Provision: The Contribution of Soft Wystems Methodology, Stowell, FA, (ed.), McGraw-Hill, London, pp. 18-50

Miles, R., Combining “hard” and “soft” systems practice: grafting and embedding revisited. *Systemist* 14 (2) 62-66. 1992;

Mingers, J., 1988, Comparing conceptual models and data flow diagrams. *The Computer Journal* 31 (4) 376-379. 1988;

Mingers, J and Gill, A (eds), 1997, Multimethodology:Towards Theory and Practice and Mixing Methodologies

Mirijamdotter, A. 1998. A Multi-Modal System Extension to Soft Systems Methodology. Doctoral Thesis, Department of Informatics and Systems Science, Lulea Technical University, Sweden, Lulea.

Mitev, N. N. 1996. More Than a Failure? The Computerized Reservation Systems at French Railways. *Information Technology and People* 9(4): 8-19
Muller, P-A., Instant UML, Birmingham:Wrox Press, 1997.

Munro, I and Mingers J., 2002, The use of multimethodology in practice – results of a survey of practitioners, Journal of the Operational Research Society, 2002, 53, 369-378.

Mumford, E. 1995. Effective Systems Design and Requirements Analysis: The ETHICS Approach. MACMILLAN Press Ltd.

Mumford, E, 1983, Designing Human Systems for New Technology: The ETHICS Method, Manchester Business School, Manchester.

Munson, J.C., 2003, Software Engineering Measurement, Auerbach Publications

Myers, M.D. and Avison, D.E. (eds.), 2002. Qualitative Research in Information Systems: A Reader, Sage Publications, London, 2002.

Myers, MD, 1994, Dialectical hermeneutics: a theoretical framework for the implementation of information systems. Information Systems Journal, 5, 51-70

Myers, M. D. 1994. A Disaster for Everyone to See: An Interpretive Analysis of a Failed IS Project. *Accounting, Management, and Information Technologies* 4(4): 185–201.

NCIHE, 1997 *Higher Education in the Learning Society: Report of the national committee*, The National Committee of Inquiry into Higher Education. Recommendation 40

Neumann, P.G., 1995, Computer Related Risks, Reading, MA: Addison-Wesley

Nielsen, J, 2005 Ten Usability Heuristics, **http://www.useit.com/papers/heuristic/heuristic_list.html** 10/02/2005

Nguyen, L and Swatman, P A, 1999 *Essential and incidental complexity in requirements models* School of Management Information Systems Working Paper 1999/15, Deakin University, Australia

Nickerson, R. S., 1999. Enhancing creativity. In R. E. Sternberg (Ed.), *Handbook of Creativity* (pp. 392-430). Cambridge (UK): Cambridge University Press.

Ormerod, R, 1995. Putting Soft OR Methods to Work:Information Systems Strategy Development at Sainsbury's, Journal of the Operational Research Society, March 1, 1995., Vol 46, Number 3, pages 277-293

Ormerod, R, 1995, The Role of methodologies in systems strategy development: reflections on experience, in Stowell, FA, (Ed.), Information Systems Provision: The Contribution of SSM, McGraw-Hill, Maidenhead

Patching, D., 1990, Practical Soft Systems Analysis, Pitman
Pidd, M (ed), 2004 Systems Modelling: Theory and Practice, John Wiley & Sons, Hoboken, N.J.

Pidd, M., 1996 Tools for Thinking: Modelling in Management Science, Wiley.

Pollice, G., 2001, *Using the Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming,* Rational Software Corporation, 2001 (at www.rational.com)

Pooley, R and Stevens, P, 1999, Using UML: Software Engineering with Objects and Components, Addison-Wesley, Harlow

Pressman, RS, 1997, Software Engineering: A Practitioner's Approach, McGraw-Hill, London

Pressman, R.S, adapted by Ince, D,1997 4th ed, Software Engineering: A Practitioner's Approach, European Adaptation by Ince, D, Mc-Graw-Hill

Prior, R., Linking SSM and IS development, 1992. *Systemist* 14 (3) 128-132. 1992

Pullum, L, 2001, Software Fault Tolerance techniques and Implementation, Artech House

Remenyi, D and Brown, A, 2002, The Make or Break Issues in IT Management: A Guide to 21st Century Effectiveness, Butterworth Heinemann

Rose, J, 2002, Interaction, transformation and information systems development – an extended application of Soft Systems Methodology, Information Technology & People, Vol. 15, No. 3, pp. 242-268, Emerald Group Publishing Limited.

Rose, J and Lewis, P, 2001, Structuration theory, action research, and information systems development, paper presented at the IFIP WG 8.2, Boise, ID

Rose, J, 2000, Information Systems Development as action research – Soft Systems methodology and structuration theory, PhD thesis, November 2002, Lancaster University, Lancaster

Rose, J, 1997, Soft Ssytems Methodology as a social science research tool, Systems Research and Behavioural Science, Vol. 14, No. 4, pp. 249-58

Rosenhead, J.V. and Mingers, J. ,2001 Rational Analysis for a Problematic World Revisited:Problem Structuring Methods for Complexity, Uncertainty and Conflict, (2nd Edition), Wiley, Chichester

Rosenwein, M. 1997. The Optimization Engine That Couldn't. *OR/MS Today* 24(4): 26-29

Sambell, K and Hubbard, A, 2004. The Role of Formative 'Low-stakes' Assessment in Supporting Non-Traditional Students' Retention and Progression in Higher Education: Student Perspectives, Widening Participation and Lifelong Learning, The Journal of the Institute for Access Studies and The European Access Network, Vol 6, No 2, August 2004

Sankaran, S, 2001, Methodology for an organisational action research thesis, Action Research International refereed on-line journal

Sarkar, P and Cybulski, J, Aligning System requirements with stakeholder concerns, 2002

Sau-Ling Lai, L, 2000, An Integration of Systems Science Methods and Object-Oriented Analysis for Determining Organizational Requirements, Systems Research and Behavioural Science, 17, 205-228

Sauer, C., *Why Information Systems Fail, a case study approach*, Waller, Henley, 1994.

Savage, A and Mingers, J, 1996, A framework for linking soft systems methodology(SSM) and Jackson system development QSD), Information Systems Journal, 6, 109-29

Sawyer, K, 1992, A contribution towards the debate on linking SSM to IS, Systemist, 14(3), 199-201

Sawyer, K, 1991, Linking SSM to DFDs: the two epistemological differences, Systemist, Vol. 14, No. 3, pp. 76-80

Schmuller, J., SAMS Teach Yourself UML in 24 Hours (2nd ed.), Sams Publishing, 2002.

Selic, B, 1999, Turning Clockwise:Using UML in the Real-Time Domain, Communications of the ACM, Vol 42, Issue 10, October 1999, pages 46-54

Senge, P, 1990, The Fifth Discipline, Random House:London

Senge, P, The Fifth Discipline: the Art & Practice of the Learning Organization, Random House Business Books, 1990.

Sharp, H, Finkelstein, A, 1999, Stakeholder identification in the Requirements Engineering Process. Proc. Database and Expert Systems Applications(DEXA 99), Florence, Italy, IEEE Computer Society Press

Skyrme, D J,1997,Multimethodologies-the Knowledge Perspective, Multimethodology:The Theory and Practice of combining management sciences Methodologies. Eds Mingers, J and Gill, A, Wiley & Sons

Si Alhir, S, 2003, Learning UML, O'Reilly & Associates, Inc

Si Alhir, S, 1998, UML in a Nutshell. O'Reily & Associates, Inc., CA

Siau, K and Halpin, T., 2001, Unified Modelling Language: Systems Analysis, Design and Development Issues, Idea Group Publishing

Smith, L W, 2000, Project clarity through stakeholder analysis: CROSSTALK: the Journal of Defense Software Engineering: 4 -9.

Smyth, D.S., and Checkland, P.B.,1976, Using a systems approach: the structure of root definitions. *Journal of Applied Systems Analysis, 5(1), 75-83*

Snoeck, M, Poelmans,S and Dedene, G, 2001, A layered Software Specification Architecture, Lecture Notes in Computer Science 1920, in Laendler, AHF, Liddle,SW and Storey, VC, ed., ER2000, 19th International Conference on Conceptual Modelling, Salt Lake City, UTAH, USA, October 2000.

Sommerville, I, 2000, Software Engineering, 6th ed., Addison-Wesley, Reading, MA

Sommerville, I and Sawyer, P, 1997, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Chichester

Spear, R, The Dark Side of the Moon – Unilluminated Dimensions of Systems Practice, Systemic Practice and Action Research, Vol 14, No. 16, December, 2001

Standish. (1995). *Most Programming Projects Are Late*. West Yarmouth (MA): Standish Group.

Stephens P. and Pooley R. *Using UML-Software Engineering with Objects and Components*, Addison Wesley, London, 1999.

Stowell, FA, 2000, Modelling IS requirements for complex systems. In Systems Modelling for Business Improvement, Bustard, DW, Kawalek, P and Morris, MT (eds), Artech House, pp. 171-86

Stowell, FA and Champion, D, 2000, Interpretivist modelling for information system definition. In Systems Engineering for Business process Change, Henderson, P (ed), Springer, pp. 106-16

Stowell, FA, West, D and Stansfield, M, 1997, Action Research as a framework for IS research. In Information Systems:An Emerging Discipline?, Mingers, J and Stowell, FA (eds), McGraw-Hill, London.

Stowell, FA. and West, D. (1994) *Client-Led Design:A Systemic Approach to Information Systems Definition,* McGraw-Hill, Maidenhead

Stowell, F and West, D, 1994, Soft Systems Thinking and information systems: A framework for client-led design., Information Systems Journal, 4, 117-127

Stowell, I. (ed.) (1995), Information Systems provision: The provision of Soft Systems Methodology – London:McGraw-Hill

Susman,GI and Evered, RD, 1978, An assessment of the scientific merits of action research, Administrative Science Quarterly, Vol. 23, pp. 582-603

Taylor, MJ, Moynihan and Wood-Harper, AT, 1998, Soft Systems Methodology and Systems Maintenance, Systemic Practice and Action Research, Vol. 11, No. 4, 1998

The Independent, June 4, 2004
http://millenium-debate.org/ind4june042.htm

Thomas, J. C., Lee, A., & Danis, C. (2002). Enhancing creative design via software tools. *Communications of the ACM, 45*(10), 112-115.

Torlak, G, 2001, Reflections on Multimethodology: Maximizing Flexibility, Responsiveness, and Sustainability in Multimethodology Interventions Through a theoretically and practically improved version of Total Systems Intervention (TSI), Ssytemic Practice and Action Research, Vol. 14, No. 3, 2001

Tsouvalis, C and Checkland P, 1996, Reflecting on SSM: the dividing line between 'real world' and 'systems thinking world', Systems Research, 13(1), pp. 35-45

Ulrich, W, 2003, Beyond Methodology Choice: Critical Systems Thinking as Critically Systemic Discourse, Journal of the Operational Research Society, April 2003, Vol 54. no 4, pp 325 -342(18)

UMISD, 1998, Unified Mechanism for Information Systems Definition

Varey, R.J, Wood-Harper, T and Wood, B., 2002, A theoretical review of management and information systems using a critical communications theory. Journal of Information Technology, 2002, 17, 229-239

Wade, S, 2004. An Approach to Integrating Soft Systems Methodology and Object Oriented Software Development, UKAIS

Wade S and Hopkins J, A Framework for Incorporating Systems Thinking into Object Oriented Design. Published in refereed proceedings of The 14th International Conference on Advanced Information System Engineering, Toronto, 2002,

Wade, S; Mansell, G and Hopkins J (2002): Integrating Systems Thinking and Object Oriented Design. OT2002 (held at Oxford University).

Walker, A, Spink, M and Vlissidis, P (1996), The Application of Structured Analysis/Formal Design Method to a Case Study from the Nuclear Industry Proceedings of the Methods Integration Workshop, Leeds, 25-26 March, Bryant, A & Semmens (Eds), Electronic Workshops in Computing, Springer

Walsham, G, 1995, Interpretive case studies in IS research: nature and method, European Journal of Information Systems, 4, 74-81

Walsham, G, 1993, Reading the Organization:metaphors and information management, Journal of Information Systems, 3, 33-46

Walsham, G, Symons, V and Warma, T,1988, Information systems as social systems: implications for developing countries. Information Technology for Development 3(3), 189-204.

Wang, Y and Bryant, A, 2002, Process-Based Software Engineering: Building the Infrastructures, Annals of Software Engineering, 14, 9-37, Kluwer Academic Publishers, Netherlands

Warmer J and Kleppe A, *The Object Constraint Language:Precise Modelling with UML*, Addison Wesley Longman, 1998.

Warmington, A, 1980 Action Research:Its Method and its Implications. Journal of Applied Systems Analysis. 7, 23-39.

Warren, L, 2003, toward Critical Intervention in Small and Medium – Sized Enterprises: A Case Study, 2003, Systemic Practice and Action Research, Vol 16, No. 3, June 2003

Webb, M and Hill, M, 2003, The Institution Gets its Act Together: linking learning and teaching and widening participation strategies to improve completion rates, Widening Participation and Lifelong Learning, The Journal of the Institute for Access Studies and The European Access Network, Vol 5, no 3, December 2003

Weisfeld, M, 2002, The Object-Oriented Thought Process: The Authoritative Solution, SAMS Publishing

Wend, P, 2004, Improving Student Success through a Joined-Up Institutional Approach, Widening Participation and Lifelong Learning, The Journal of the Institute for Access Studies and The European Access Network, Vol 6, No 2, August 2004

West, D, 1995, The appreciative inquiry method: a systemic approach to information systems requirements analysis. In Information System Provision: The Contribution of Soft Systems Methodology, Stowell, FA (ed), McGraw-Hill, London, pp. 140-58

Willocks, L and Graeser, V, 2001, Delivering IT and E-Business Value, Butterworth Heinemann

Wilson, B, 2001 Soft Systems Methodology: Conceptual model building and it s contribution, Wiley Publishers

Wilson, B., 1990, *Systems: Concepts, Methodologies, and Applications*, 2nd Edition, John Wiley & Sons, New York, 1990

Winter, MC, Brown, DH and Checkland, PB, 1995, A role for soft systems methodology in information systems development, European Journal of Information Systems, Vol. 4, pp. 130-142

Wood-Harper, A.T. "Viewpoint: Action Research," *Journal of Information Systems* (2), 1992, pp.235-236.

Wood-Harper, A.T. "Research Methods in Information Systems: Using Action Research," in *Research Methods in Information Systems*, E. Mumford, R.A. Hirschheim, G. Fitzgerald and A.T. Wood-Harper (eds.), North-Holland, Amsterdam, 1985.

Xu, L.D, 1995, Systems Thinking for Information Systems development. Systems Practice 8(6), 577-589.
Zachman, JA, 1987, A framework for information systems architecture, IBM Journal 26(3), 1987, 276-292

Zhao, J, 2004, Robust Object Oriented Systems Analysis
http://consulting.dthomas.co.uk

URLs

http://www.hefce.ac.uk
http://plw.media.mit.edu/people/maeda/designmsthesis.pdf, Maeda, John, 2002
http://www.ctg.albany.edu/publications-19/01/2005 - Centre for Technology in Government
http://www.sdmagazine.com, 2005