3-31-2009

A Systems Thinking Approach to Domain-Driven Design

Mohammed Salahat
Ajman University of Science and Technology, UAE & University of Huddersfield, UK, abac.hasan.m@ajman.ac.ae

Steve Wade
Informatics Department, School of Computing and Engineering, University of Huddersfield, UK, s.j.wade@hud.ac.uk

A SYSTEMS THINKING APPROACH TO DOMAIN-DRIVEN DESIGN

Mohammed Salahat
Ajman University of Science and Technology, UAE
abac.hasan.m@ajman.ac.ae, *& University of Huddersfield, UK* m.salahat@hud.ac.uk

Steve Wade
Informatics Department, School of Computing and Engineering, University of Huddersfield, UK s.j.wade@hud.ac.uk

Abstract

This paper builds on our earlier work in the construction of a systemic framework for developing information systems. In this paper we apply the framework to the development of a Peer-Tutoring System (PTS) for Introductory Programming courses in our Universities. The framework supports the full development life cycle from business process modelling to software implementation. We use Soft Systems Methodology (SSM) as a guiding methodology within which we have embedded a sequence of design tasks based on the Unified Modelling Language (UML) and Domain-Driven design techniques h Naked Objects Pattern is used as DDD approach. This leads to the implementation of a prototype software application using the Naked Objects framework. We have involved developers and management in reviewing the software system and the approach taken to develop it. The results suggest that the framework can lead to improved business process modelling and software implementation.

Keywords: Peer-Tutoring, Workflow, SSM, UML, Naked Objects, Multimethodology, Domain-Driven Design

1.0 Introduction

One of the main reasons for information systems failure is a tendency to concentrate on the technical aspects of design rather than understanding the business needs (Alter, S, 2007). This suggests a need to bridge the gap between business process modelling, information systems modelling, and implementation. There is a need for a systematic framework or methodology to explore all issues related to the problem situation, and to capture the information required by business processes (Sewchurran, K. & Petkov D, 2007).

A number of software systems development methods have been widely used since the seventies. Some of these, such as SSADM (Structured Systems Analysis and Design Method) (Ashworth and Goodland, 1990) have a reputation for being bureaucratic and have not been generally popular with programmers. This is partly because the inspiration for such methods has come from engineering disciplines such as civil or mechanical engineering. These disciplines put a great deal of emphasis on the need to spend a lot of time planning before you construct anything. The engineering approach is characterised by work on a series of models that precisely indicate how a software system should be constructed. This approach can be attractive to management because it allows for the identification of tasks that need to be carried out and of the dependencies between these tasks, suggesting the possibility of a predictable schedule and budget for systems development. A key argument against this approach is that it encourages the project manager to plan out a large part of the software development process in great detail for a long time ahead, this makes both the approach and the software developed using the approach, resistant to change.

In more recent years there has been a great deal of interest in lightweight or “agile” methods that attempt to compromise between no development process and an overly prescriptive process, providing “just enough” process for a given project (Ambler, 2002). Agile methods have been heavily influenced by the rise in popularity of object-oriented programming languages, such as C# and Java supported by object-oriented and object-relational databases. These tools allow programmers to develop software solutions quickly, hence the reduced need for detailed design steps in the development process. The ubiquity of object technology at the programming level is represented at

the design level by the Unified Modelling Language (UML) (Fowler and Scott, 2000) which has been widely adopted as a standard notation for software design.

The UML defines a number of diagrams that can be used to describe an evolving software system; it does not describe a method for actually building the software. A number of development methods have been proposed that use the UML with varying degrees of agility. Amongst the least agile of these the Unified Software Development Process (USDP) (Jacobson *et al*, 1999) and the Rational Unified Process (RUP) (Kruchten, P., 2000) have attracted a great deal of attention. Amongst the more explicitly agile methods, Alistair Cockburn's Crystal family of methods (Cockburn, 2002), Jim Highsmith's Adaptive Software Development methods (Highsmith, 2001) and Peter Coad's Feature Driven Development (Coad, 2002) have been influential.
Agile methods are usually documented in terms of a base method that can be tailored on a project-by-project basis. The process of configuring the base method involves comparing a conceptualised model of a generic software development process with the specific technical and cultural requirements of a particular project. The use of a conceptualised model in this way resonates with a popular approach to analysing systems known as Soft Systems Methodology (SSM).

SSM (Checkland, 1981; Checkland and Scholes, 1990; Checkland and Howell, 1998) is an established means of problem solving that focuses on the development of idealised models of relevant systems that can then be compared with real world counterparts. The approach can be applied in a wide range of situations including requirements analysis for information systems design. The majority of work in this area relates to attempts to integrate SSM with the type of structured development methods that preceded object oriented technology (Mingers, 1988; Avison and Wood-Harper, 1990; Keys and Roberts, 1991; Miles, 1992; Prior, 1990; CCTA, 1993; Stowell and West, 1994). Some researchers have explored the relationship between SSM and object oriented analysis and design techniques in general (Bustard, D *et al,* 1996; Lai, L.S. 2000) but less has been written about the application of these techniques in the context of the UML.

However agile they may be, all modern development methods recognise that business software requirements are highly volatile. In the past there was a tendency for

methodologists to address this problem by spending a long time obtaining a detailed picture of requirements and then getting the customer to sign-off to these requirements before proceeding to the design and construction phases. This approach is flawed because users increasingly find themselves in changing business situations and are therefore unable to identify unalterable requirements. The model of software development as an adaptive process, in which detailed requirements emerge iteratively as a project progresses and are modified as learning takes place, seems much more appropriate. There is however a problem with this approach because all other software tasks are driven by requirements. If we cannot get stable requirements we cannot get a predictable plan. This raises the question of how we might exert some control over unpredictability. The response to this question, adopted by virtually all modern development methods, has been an increased emphasis on "use cases" and "iterative" development techniques.

A "use case" might be defined as a piece of functionality that provides meaningful value to a user. For example, "check spelling of selected word" might be a suitable use case for a word processor. In an iterative approach, development is organized into a series of short, usually fixed-length (for example, four-week) mini-projects called "iterations". The outcome of iteration should be a tested, integrated and executable system that delivers a subset of the required features of the whole system. Specific iterations are likely to relate directly to a group of closely related use cases.

We argue that there are certain types of project where requirements are so unclear that the use case approach is insufficient as a means of identifying suitable iterations. The conclusion that techniques from Soft Systems Methodology (SSM) should be added to the developer's armoury is in keeping with the pragmatic nature of agile development methods. We have reached this conclusion by reflecting on our own experiences of developing information systems to support the activities of the schools in which we are employed.

The key aim of the research discussed in this paper has been to investigate ways of integrating techniques from SSM (Soft Systems Methodology) into the requirements elicitation stage of an agile system development method based on the UML (Unified Modelling Language) and techniques from Domain Driven Design (Naked Objects).

We argue that used alone UML models can encourage early design decisions before opportunities for improvement have been agreed and that SSM lacks the detailed information required by designers developing domain models. This leads to the conclusion that there could be some advantage in using the techniques together.

In developing an integrated method we have been influenced by the recent trend towards agile systems development. This represents a move away from seeing software development methods as codified practices focusing on specific artifacts within a prescribed lifecycle. Instead emphasis is placed on the provision of a framework of development activities, products and workflows together with guidance for applying these to a particular application area.

2.0 Research Methodology

This research aims to answer the following research question:

How can we formulate a multimethodology framework that combines soft and hard organizational models in order to model, design, and implement internal business process in a workflow system?

The design of the research is as follows:

1. A series of Information Systems Development (ISD) projects are being carried out using SSM, UML and Domain-Driven techniques to make recommendations about the design of our School's intranet. These are being written up as case studies.
2. The domain models developed in these case studies are being used in the development of prototype applications using the Naked Objects Framework. These applications are then being evaluated for usability.
3. The case studies are being used to reflect upon and develop a hybrid method (or development framework) and a supporting CASE (Computer Assisted Software Engineering) tool also developed using the Naked Objects Framework.

To answer the above research question and to apply the research as it's designed, the following methodology followed:

1. Review the current situation of business process and workflow modelling, design, and implementation status.

2. Compare and construct business process and workflow modelling, design, and implementation approaches.

3. Formulate and propose a multimethodology framework considering soft and hard organizational aspects.

4. Evaluating the framework through different practical case studies (Peer-Tutoring System, Work Placement Operations Mgt. System, and University Students Associations System). This will be done by performing the following operations on all case studies:

a-Explore the business situation problems using SSM as a guiding methodology.

b- Model the business process as a workflow system using UML

c- Design and Implement the workflow system using DDD (Ii.e Naked Objects Pattern is selected)

5. Reflect on the implementation and record learning from the methodology application in order to guide further applications.

The discussion of how the proposed framework emerged from our practical experience is punctuated with UML and other types of model that relate to the design of the supporting CASE tool.

3.0 The Systemic Soft Workflow Modeling and Implementation (SSWfMI)

SSWfMI framework is developed in our previous work (Salahat et al, 2008) and it can be used to investigate the problematic situation, model, design, and implement any system required a deep investigation and lead to practical software solution. SSM will be applied first to investigate the problematic situation, UML will be used to model and design the system, and Naked Objects Framework will be used for implementation. The framework consists of four phases: Pre-SSM, SSM, Post1 SSM, and Post2 SSM. This framework is different from others since it's the first framework combined soft and hard system concerns with a complete system life cycle up to the implementation. As stated before, many systems failed because of a lack of detailed investigation for both hard and soft systems concerns. It is essential to identify the changes required for the investigated domain before starting further stages which may

lead to inappropriate implementation. The proposed framework avoids these problems and tries to explore the investigated systems properly. For more details about the adapted model see (Salahat et al, 2008). The framework represented in Figure (1) and Figure (2) is a flowchart showing the logical processes embedded in the framework.

In Section3 we will discusses the experience of applying this approach in two case studies.

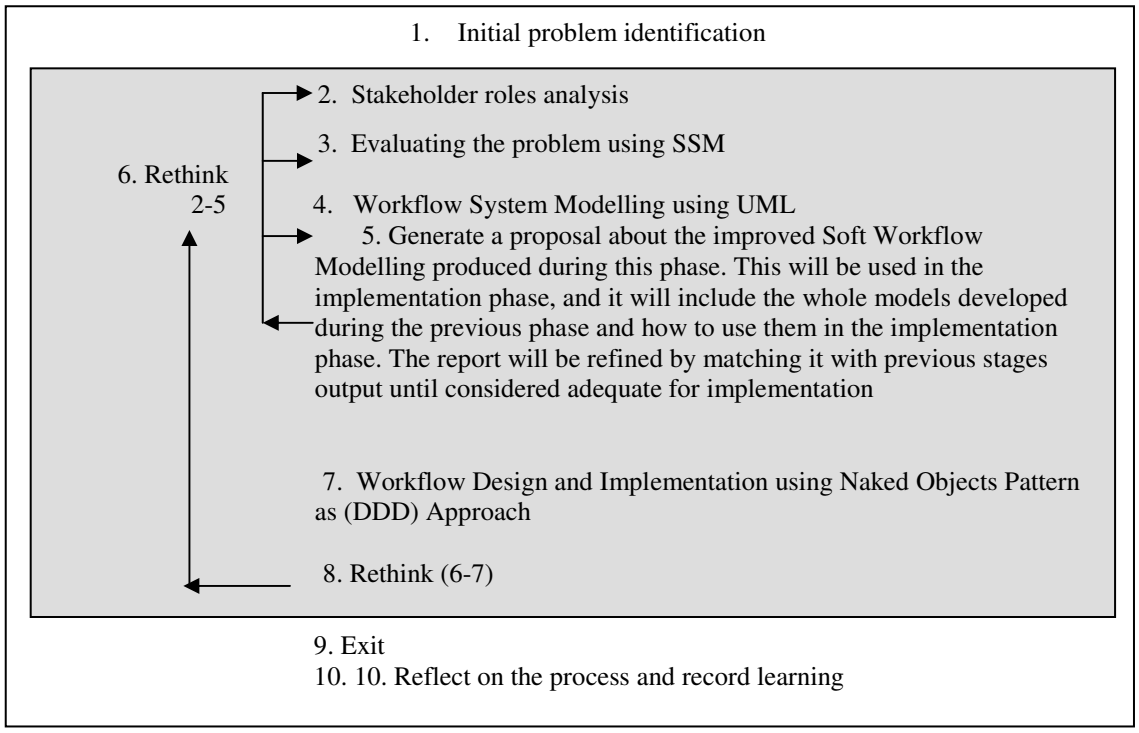


Figure 1: Systemic Soft Workflow Modelling and Implementation

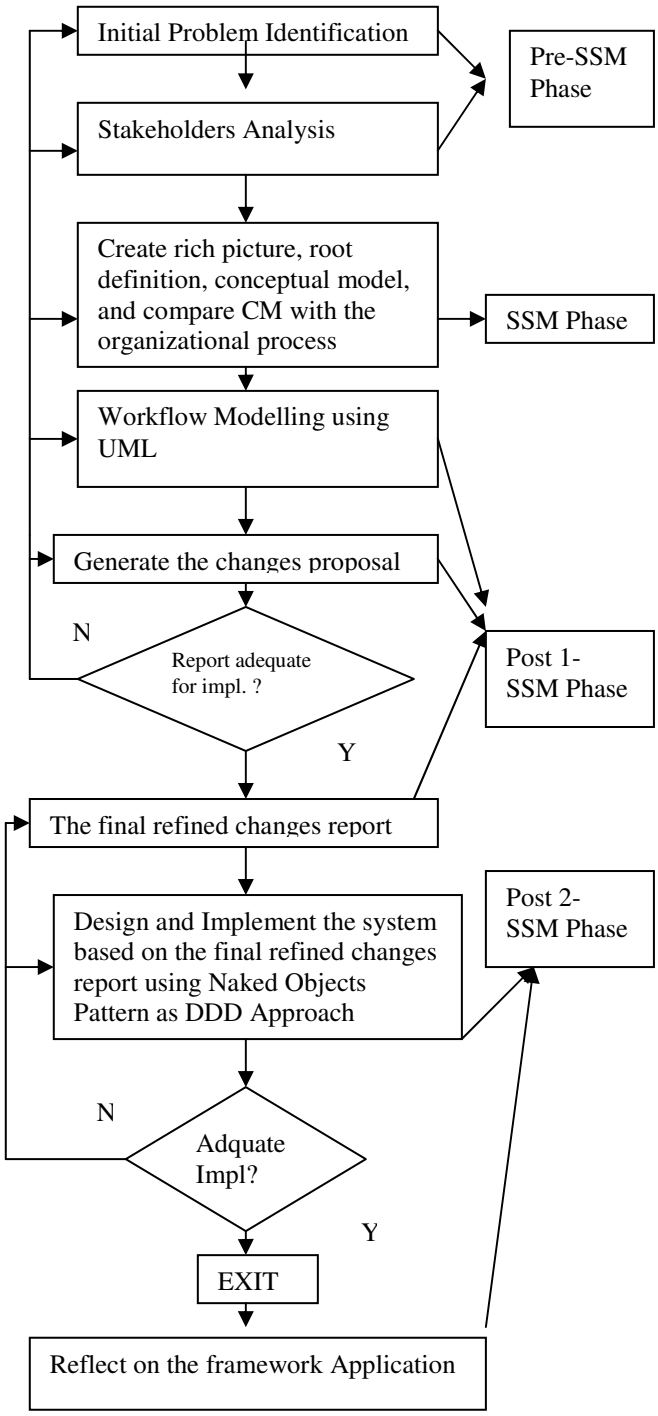


Figure 2: The logical processes in the framework

4.0 The Case Studies

We have been engaged in an information systems development project using SSM and UML techniques within an agile framework to make recommendations about the development of an intranet for the academic school in which we are employed. At the beginning of the project the department had an operational intranet but this was not widely used. An information system strategy was initiated to investigate ways in

which the intranet could be developed to support the university mission and departmental goals. Initially we used use cases as the primary fact-gathering technique but certain limitations in this approach led us to a more thorough SSM-based analysis of the situation.

We argue that the techniques of SSM can help the developer to identify a richer set of use cases than would otherwise be possible but developers with a full use case model still have many challenges ahead of them. We are interested in object oriented design and the view that all business behaviour identified in the use case model should be encapsulated as methods on domain objects. Thus, a Student object should not just be a collection of data about the Student; it should encapsulate all the behaviours that we need to apply to a student. In Domain-Driven Design these are often referred to as 'behaviourally-rich' domain objects.

A number of software frameworks have been developed to allow programmers to build prototype applications directly from a behaviourally rich domain model implemented in an object oriented programming language. Prominent amongst these is the Naked Objects Framework. This is the one that we have chosen to use to implement our prototype applications.
In the next section we present a quick superficial description of how the method might be applied to a relatively simple project, the design and implementation of a peer-tutoring system.

4.1 A Peer-Tutoring System

One of the current problems facing students and lecturers in university is the difficulty of understanding and mastering the skills required to write and run computer programs successfully. A number of researchers have suggested that peer tutoring can be particularly useful to support this type of learning because it allows learners to learn and support each other (Goodlad and Hirst, 1989), and it is beneficial to help students learn and practise the required skills more actively in a setting that encourages them to be more active and intellectually engaged (Gardner 1993).

Iwona Miliszewska and Grace Tan (2007) reported about the problems of teaching programming course at Victoria University in Australia and they proposed an approach to enhance the delivery of this module.
Hu Xiaohui (2006) raised the difficulties of teaching programming course in Chinese universities and discussed different modern incorporating strategies, to solve this problem, which includes "Concept Mapping", "Peer-learning" and "E-learning" methods.

The proposed solutions to recap the difficulties of teaching programming unit by the mentioned researchers concentrating on the delivery methods only without investigating all soft and hard systems issues that can cause such a problem(Hu Xiaohui ,2006 , Iwona Miliszewska and Grace Tan, 2007). In this work, we proposed Peer- tutoring system as an improvement of the teaching process and to enhance the students understanding which may be reduce the percentage of failures. In the next sections we will show how the method is applied.

4.1.1. Pre-SSM Phase

The problem identification

The Department of Informatics in the School of Computing and Engineering at the University of Huddersfield in UK and Information Technology College at Ajman University of Science and Technology in UAE both offer introductory programming modules for their first year computing students. These modules focus on Java programming; lecturers face certain difficulties related to students understanding of the subject because of the nature of the required problem-solving skills. Students require more tutoring and practical sessions to help them practise different exercises in order to enhance their understanding and practical skills. Both Universities expect that implementing a peer-tutoring system will reduce the failure rate. The departments want to know how to select tutors among good students and how to reward them.

Stakeholder Determinations

The stakeholders of the required system were determined to be peer tutor, peer tutee, lecturer, and management. The stakeholders have different expectations of the system. Peer tutors are generally looking for teaching experience to be added to their CVs.

Peer tutees are looking for extra help. Lecturers are looking to reduce their workload, and to determine which students most require tutoring sessions. Management look to reduce the number of failures on programming modules.

4.1.2 SSM Phase

Investigating the problem situation using a rich picture

In order to develop a rich picture of the situation under study, a number of information sources were used to capture views of the introductory programming unit from the perspective of the management (the school & the college in both universities), lecturers, and students. Interviews with the school (or college) administration and groups of students were conducted to understand the problematic situation of teaching introductory programming course and set out suggestions to solve the problems. Rich pictures were used as a tool used in this investigation. A **number of different pictures were drawn the following is a simple early example.**

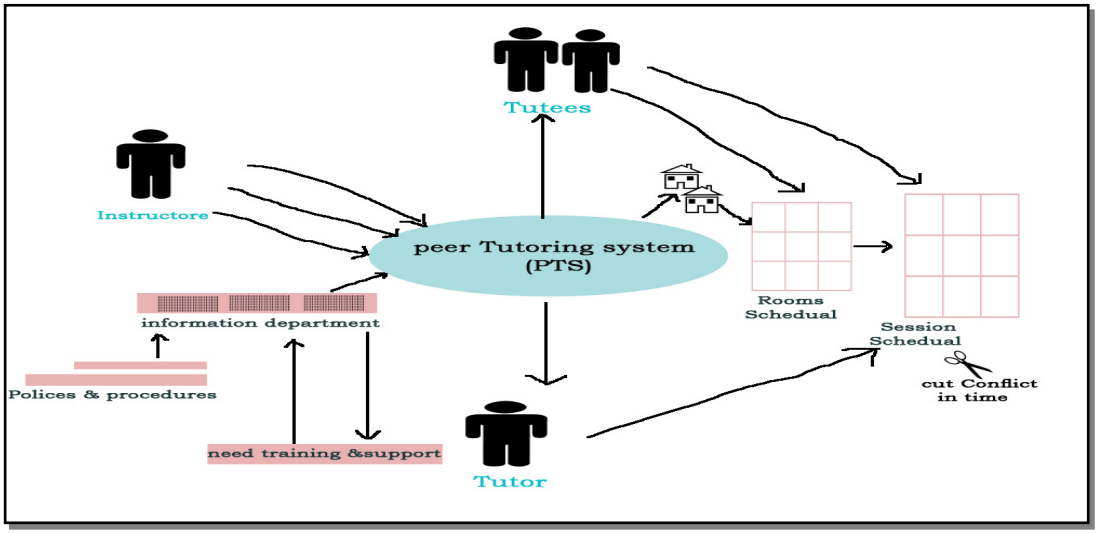


Figure (3) Peer-Tutoring System Rich Picture

Modelling the relevant system

The relevant system was modelled using a root definition and conceptual models. Our initial root definition was as follows:

"a peer-tutoring system for the informatics department will help in the selection of peer- tutees and peer-tutors, the scheduling of tutoring sessions based on the availability of rooms, tutors, and tutees. The system will also monitor the perceived

benefit to tutors and the progress of tutees in increased self-confidence as well as measure the impact on failure rates."

A variety of conceptual models were then developed to model the key activities in the system. From these a simple Consensus Primary Task model (CPTM) was developed identifying the core activities that the first version of the system would need to suppot.

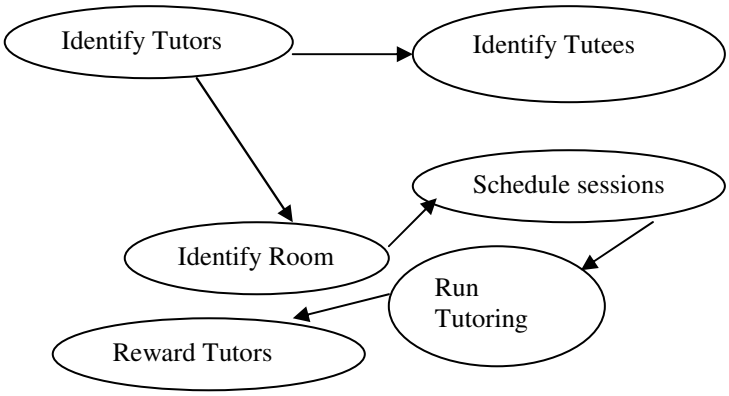


Figure (4) CPTM of Peer-tutoring System

Compare the conceptual model to the real world:

SSM required the investigator to compare the produced conceptual model with the actual real life work. There is no real life PTS available to be compared with the developed conceptual model. In this case, the conceptual model will be considered the base to model the PTS system as a workflow system as indicated by other related work (Al Humaidan, F, 2006). The CPTM, as a combination of all conceptual models, will be used in the next phase for modelling, design, and implementation of PTS as a workflow system using Domain-Driven Design approach. Naked Objects will be used as a DDD approach for this purpose.

4.1.3 Post1- SSM Phase

Workflow modelling using UML

This section consists of three parts: converting CPTM into use cases, use case modelling using UML, and Class diagram development.

Converting CPTM into use case

Any activity required software support will be selected as a use case. The stage of moving from an SSM conceptual model to a use case model is not as straightforward

as this high-level discussion would suggest. In thinking this through we have been pushed towards making a clear distinction between stakeholder goals, business activities and use cases. The following model (Figure 5) shows the relationship between these key abstractions:

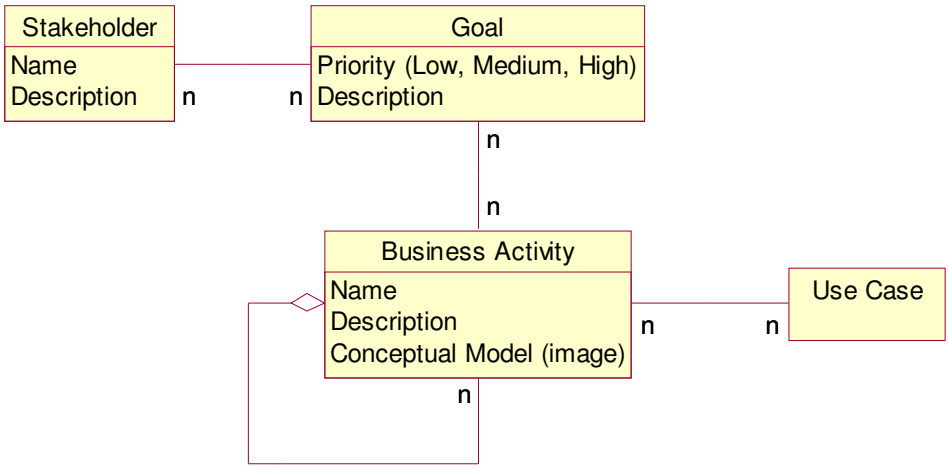


Figure (5) Moving from an SSM conceptual model

The model suggests a hierarchy of business activities related to stakeholder goals that are taken to be the primary reasons for developing the system. The business activities would be represented in a hierarchy of conceptual models with the lowest models containing more primitive, elementary business activities than the higher ones. An individual business activity is represented in context in the image of the conceptual model of which it is a part. Some of the determined use cases are presented in the following **Use Cases Diagram (Figure 6):**

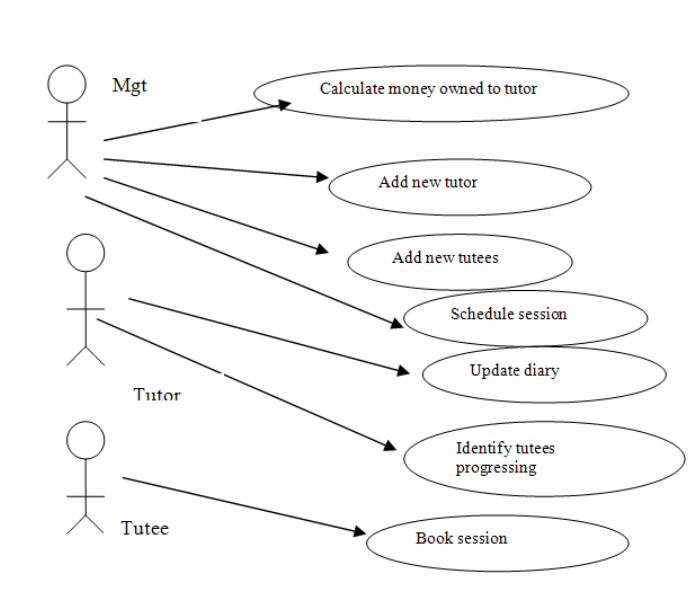
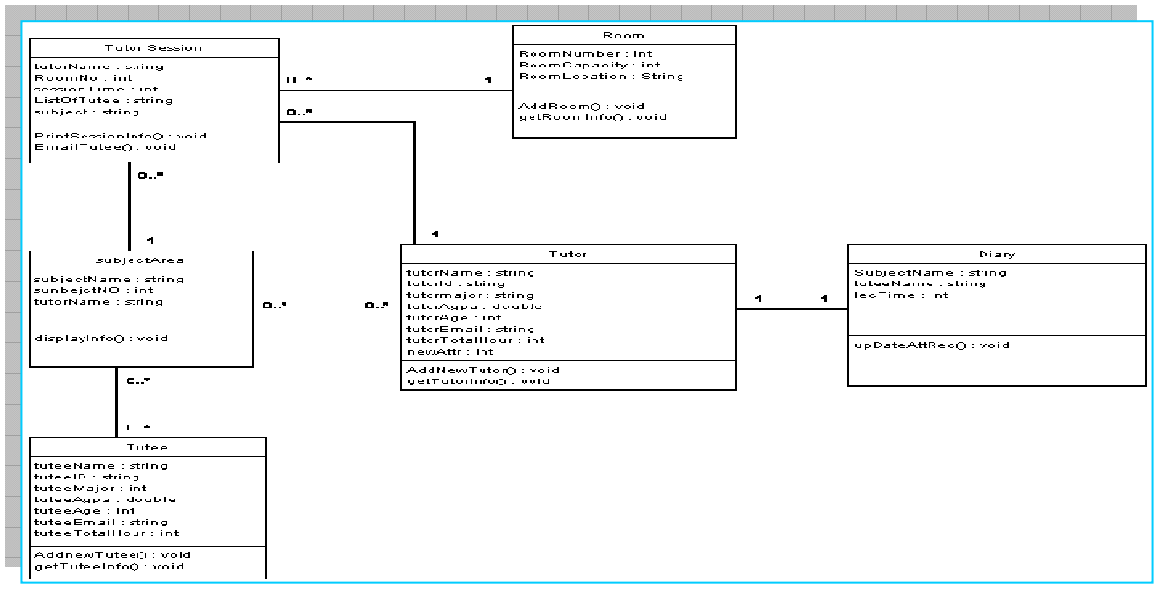


Figure 6: Use case diagram

Developing the class diagram of PTS

Each use case presented using textual template, activity diagram, sequence diagram, and all use cases are combined in a use case diagram. The next step in the process is to take the business logic identified in the use cases and associate it with classes in a class diagram. We have followed the guideline that all important business logic must be implemented in classes in the domain model. An initial class diagram is presented below. (Figure 7)



Figure(7) PTS Class Diagram

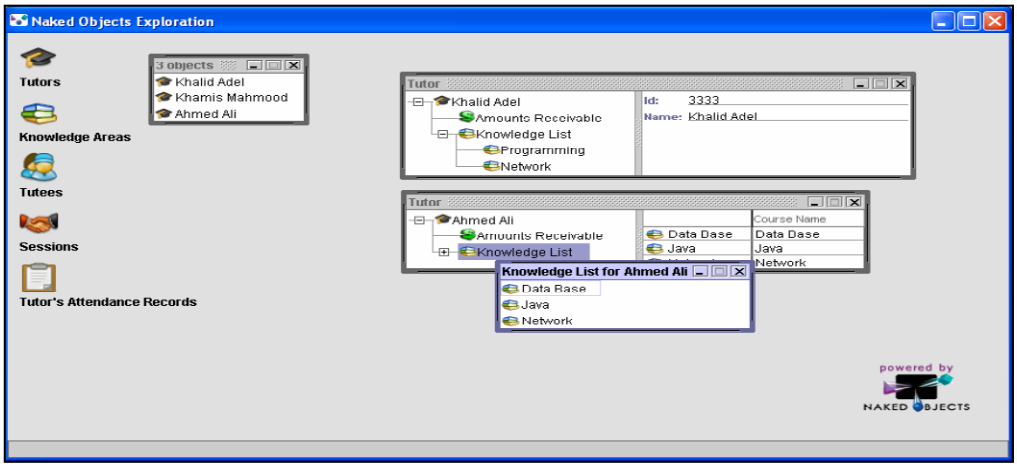
Change report generation and refinement

As shown in the framework (SSWfMI), there is a draw back to the previous stages to refine what's done during Pre-SSM, SSM, and Post1-SSM. This refinement is essential to be sure that the exact changes required already modelled well as a workflow system. As a guiding methodology, SSM focus on the generation of the required change report as a result to be recommended for the management for further actions (Checkland, P., and Poulter J, 2006, Checkland, P., 1999, Checkland, P. and Holwell, S.E ,1998). SSWfMI extended SSM further step to include implementation as a major action to be taken as part of the improvement change to enhance the investigated situation. This indicate that the implementation will be started after the completion and the refinement of the change report (includes the workflow model) to facilitate the implementation process and eliminate the possibility of system failure since all soft and hard system concerns are investigated, modelled, refined, and included in the workflow system for implementation.

4.1.4 Post2-SSM Phase

Prototype Design, Implementation, Refinement

The class diagram is used to design the domain objects which lead to a domain model which was implemented in Java and the Naked Objects framework, As DDD, was used to generate an initial prototype where the interface allows users to interact directly with the domain objects. A screenshot is provided below to give an idea of what the initial prototypes looked like: (Figure 8)



Figure(8) Naked Object Screenshot from PTS Prototype

More improvement and work is going on to enhance the productivity of the prototype to be a real system. Currently, we are Naked Objects .Net to get a real live software product, and may domain-driven design features added to this version. The new output of the current work and further enhancement on the proposed framework will be a target of a new publication.

4.2 The Placement Unit

In the previous case study (PTS) we presented detailed about the application of the proposed framework to real case studies. In this case study the work is still going on and we will present it shortly as other parts still to be completed.

Many of our courses include a twelve-month industrial placement which needs to be carefully integrated into the curriculum. This is only possible when the placement is well-managed incorporating assignments that promote self-assessment and personal development. The MaPPiT system was developed to support this. The system has been developed to support the following root definition:

A system owned by the placement unit to secure, develop and monitor rich learning experiences (on placement) that build on students' current skills and knowledge, in line with their career aspirations; enhance their employability through experience in the workplace; and increase their skills and knowledge, subsequently enabling higher levels of achievement.

An initial conceptual model developed from this root definition included the following high-level activities:

- liaise with placement providers
- prepare students for placement
- find and vet placements
- match students to placements
- plan the placement programme
- monitor the placement

- help employers to supervise and appraise the placement
- equip students to reflect on and analyse the placement learning
- assess/accredit the placement achievements

Each of these activities was then decomposed into a more detailed conceptual model containing more specific business activities in the hierarchical manner suggested earlier. For example the process "liaise with placement providers" is concerned with looking after liaison with key companies. Some companies will be important to the Placement Unit, and the unit will have a greater knowledge of these companies and what they are looking for. It is recommended that the unit should proactively seek suitable students for these companies in order to maintain the relationship with them. This process is represented as a conceptual model below.(Figure 9).

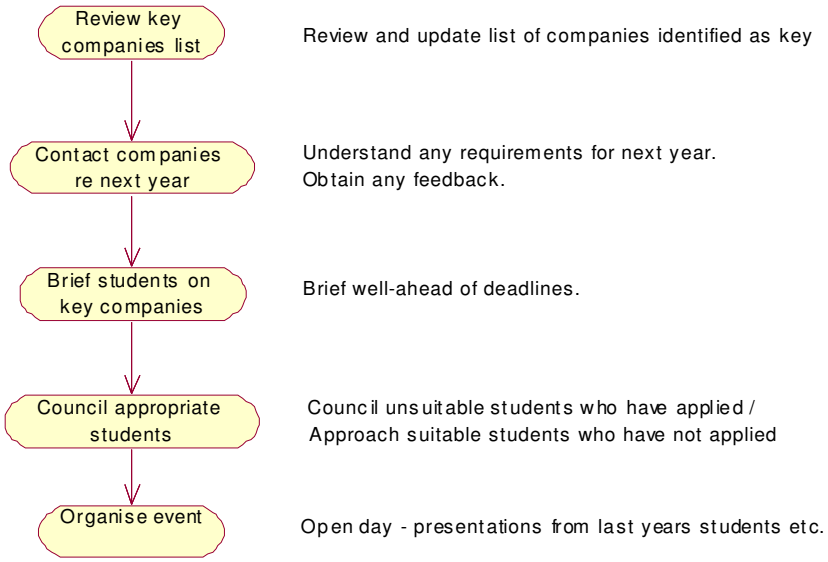


Figure (9) Work placement Unit Management System Conceptual model

Once this type of model had been developed for each of the key processes identified above we had a clear understanding of the problem situation and were able to identify some concrete use cases (e.g. "retrieve key company records", "email key companies" etc.) and domain classes (e.g. company, student etc). A system developed from this analysis is currently being developed and should form the basis for a more detailed evaluation of the method than that presented here.

5.0 Issues in documenting and supporting the proposed framework

In the examples presented above we have come close to prescribing a step-by-step procedure for converting relevant parts of root definitions and SSM conceptual models into use case models. The diagram below communicates an idea of how this step-by-step process is currently conceived (Figure 10).

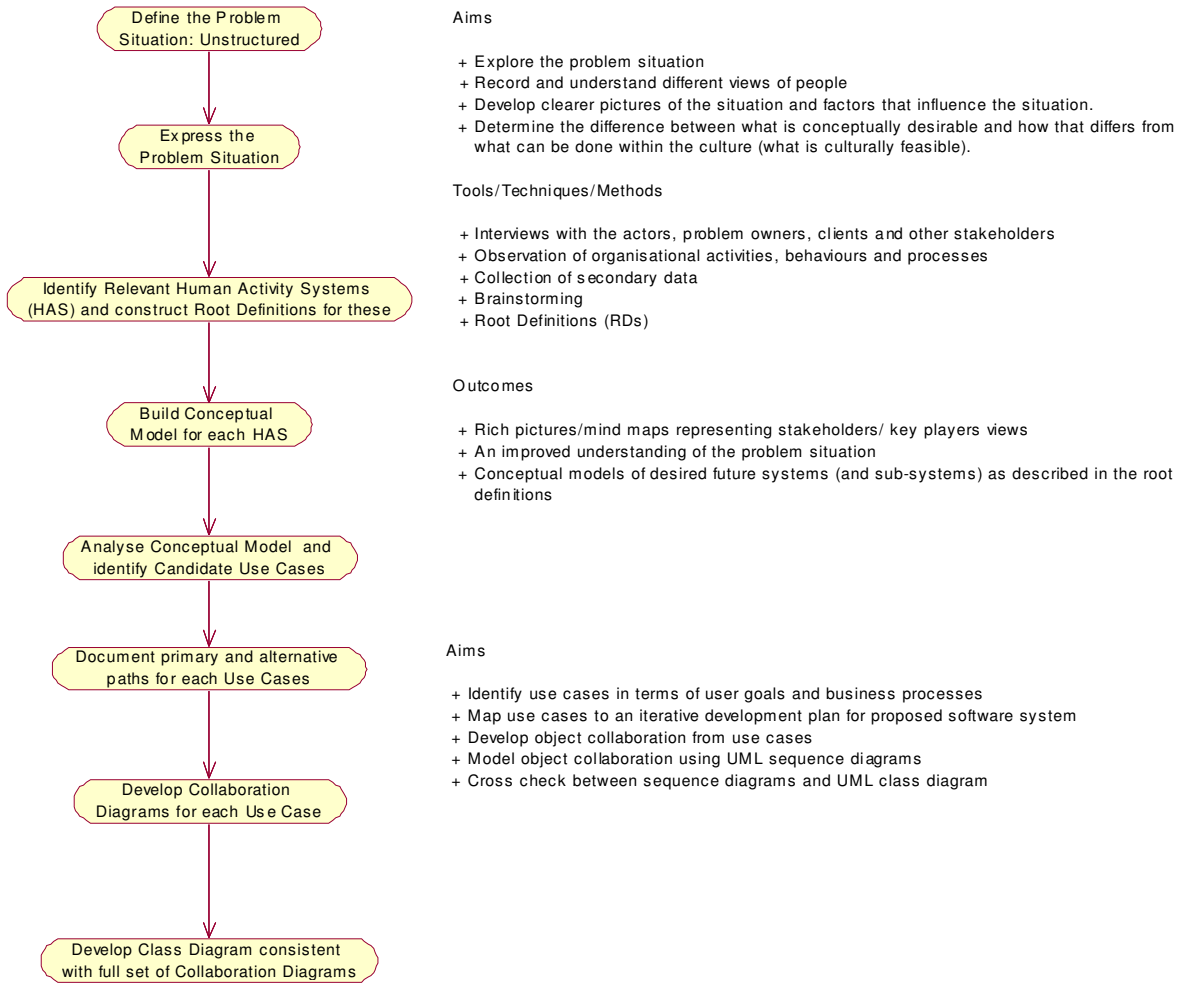


Figure (10) Step-by-step moving from SSM to Use Cases

Presented in this way the method seems prescriptive but this is not the intention. The above diagram should be interpreted as an SSM conceptual model. The appropriateness of this model should be discussed on a case-by-case basis. For example for each activity we should ask, with respect to a specific project or iteration within a project, the following questions: How will this activity help to meet the goals

of this project/iteration? How will I assess the impact that this activity is having on the achievement of those goals? It is anticipated that the entire method would only be applied in situations characterised by uncertainty and confusion at the outset.

In an attempt to support our framework of techniques we have been developing a simple CASE tool that does not impose a specific step-by-step method. The following diagram gives an idea of the principle abstractions that will be manipulated by the tool and how they are related, Figure(11).

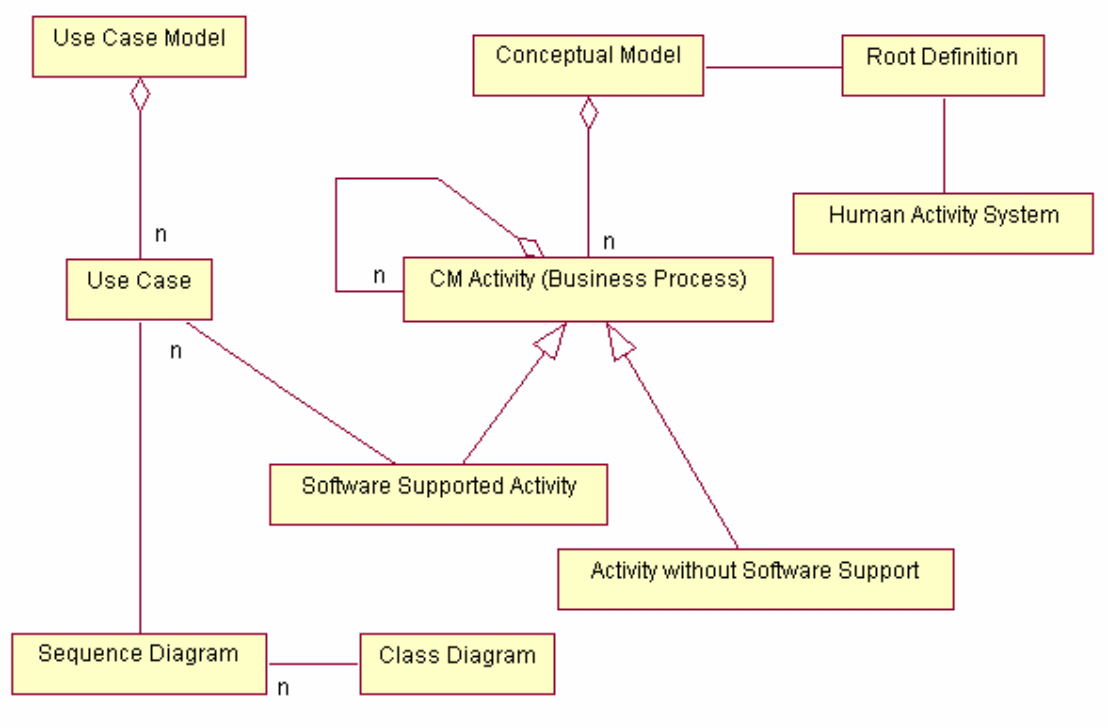


Figure (11) Proposed Case TOOL to support the framework

At present we have a prototype tool in which data about these abstractions and the relationships between them are held in an MS Access database. We want to develop this tool into a Naked Objects application that will allow the user to explore the relationships between various classes. For example we should be able to select a use case and see how it is supported by the behaviourally-rich classes we have identified in our class diagram.

The concept of"iteration" is not represented in the above model but we might expect the choice of iterations to be influenced by earlier identification of relevant systems.

For example software to support our “peer tutoring system” could be developed in a single iteration. Development of software to support more complex system (such as the “industrial placement system”) would be accomplished through a series of closely related iterations.

In contemplating the figures above some people may be concerned about the highly participative nature of our approach and the demand for documentation which can make it very time-consuming. It has been argued that web-based software systems should be developed in a software culture that is simpler, faster and more responsive to users than the one suggested here (Beck, 2000). The argument is concerned with our requirement for a large up-front commitment. In the full version of the method, stakeholders must engage in lengthy discussions based on SSM techniques and be interviewed by process experts who are able to develop formal use cases, from which the developer can produce UML class and collaboration models. The choice between unmanaged chaos and over-managed process is a long-standing one in software design. We argue that in situations such as the one discussed here, where the benefits of developing an intranet are unclear and possibly unquantifiable, linking the development process to fundamental business activities is self-evidently important.

5.0 Conclusion

The key aim of the research discussed in this paper has been to evaluate our previous proposed and published framework(SSWfMI) which integrated techniques from SSM (Soft Systems Methodology) into the requirements elicitation stage of an agile system development method based on the UML (Unified Modelling Language) and techniques from Domain Driven Design (Naked Objects Pattern) for business process modelling and implementation as a workflow system.. We argue that used alone UML models can encourage early design decisions before opportunities for improvement have been agreed and that SSM lacks the detailed information required by designers developing domain models. The work presented the evaluation results through the development of two real systems (Peer-Tutoring System, Work placement Management System which is still going on). This leads to the conclusion that there should be some advantage in using the technique together. To support the framework,

a CASE tool has been developed and the work is going on to present it in a Naked Objects application that will allow the user to explore the relationships between various classes. Further applications (University Students Associations System and Module Selection System) are started to have further improvement and refinement of the proposed model. All systems selected from our environments as an action research required to apply the framework.

References

Al Humaidan, F., (2006) *Evaluation and Development Models for Business processes",* PhD thesis, University of Newcastle, UK.

Al-Humaidan, F., & Rossiter, N. (2204) *Business Process Modelling with OBPM combining soft and hard approaches*, in Proceeding of 1st Workshop on Computer Supported Activity Coordination (CSAC), 6th International Conference on Enterprise Information Systems, Porto, , pp 253-260.

Alter, S., (2007) *The work system method: Connecting people, processes and IT for business results"*, Work System Press, Larkspur, CA.

Ambler, S.W. (2002) *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. John Wiley & Sons

Ashworth, C. and Goodland, M. (1990) *SSADM: A practical approach*. McGraw-Hill.

Avison, D.E. and Wood-Harper A.T. (1990) *Multiview: An exploration in Information Systems Development*. Blackwell Scientific Publications.

Beck, K. (2000) *eXtreme Programming Explained***.** Addison Wesley, 2000

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, pp. 52-59.

Carroll, M. (1996) *Peer Tutoring: Can Medical Students Teach Biochemistry?* Biochemical Education, 24, 13-15.

CCTA (1993) *Applying Soft Systems Methodology to an SSADM Feasibility Study*. HMSO, London.

Checkland, P. and Holwell, S.E. (1998) *Information, Systems and Information Systems, Making sense of the field*, John Wiley and Sons Ltd, West Sussex, England.

Checkland, P., (1999) *Systems Thinking, Systems Practice*, John Wiley and Sons Ltd, West Sussex, England.

Checkland, P., and Poulter J. (2006) *Learning for Action. A short Definitive Account of Soft Systems Methodology and its use for Practitioners, Teachers and Students*", John Wiley and Sons Ltd, West Sussex, England.

Checkland, P., and Scholes (1990) J, *Soft Systems Methodology in Action*, John Wiley & Sons, New York,

Coad P., et al (2002) Java Modeling In Color With UML: Enterprise Components and Process Prentice Hall

Cockburn A. (1997) Structuring use cases with Goals. *Journal of Object Oriented Programming.* Sep-Oct and Nov – Dec. SIGS Publications.

Cockburn A. (2001) *Writing Effective use cases*. Addison Wesley.
Cockburn, A. (2002) *Agile Software Development*. Addison Wesley Professional
D. Georgakopouls , M. Hornick, and A. Sheth. (1995) *An overview of Workflow Management: From Process modelling to workflow Automation Infrastructure*, Distributed and Parallel Databases, vol. 3, Springer, online publishing, pp. 119-153.
D. Platt. (1994) *Process Modelling and Process Support Environment to Design Management*, Department of Civil Engineering, Faculty of Engineering, University of Bristol, UK.
Dunican, E. (2002) Making *the analogy: Alternative delivery techniques for first year programming courses*. In J. Kuljis, L. Baldwin & R. Scoble (Eds), Proceedings from the 14th Workshop of the Psychology of Programming Interest Group, Brunel University, 89-99.
Erikksonn, H. E., & Penker, M. (2000) *UML business process modelling at work,* John Wiley and Sons, New York.
Fowler, M. and Scott, K., (2000) *UML Distilled*. Reading, M.A.: Addison-Wesley.
Gardner, H. (1993) *Multiple intelligences: the theory in practice*. New York, NY:Basic Books.
Goodlad, S. and Hirst, B. (1989) *Peer Tutoring: A Guide to Learning by Teaching,*London: Kogan Page; New York: Nickols Publishing.
Highsmith, J. (2001) *Agile Software Development Ecosystems*. Wiley.

Höysniemi, J., Hämäläinen, P., & Turkki, L. (2003). Using peer tutoring in evaluating the usability of a physically interactive computer game with children. Interacting with Computers, 15, 203-225.
Hu Xiaohui. (2006) *Improving teaching in Computer Programming by adopting student-centred learning strategies, China papers, issue 6. 46-51.*
Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*. Addison-Wesley.

Jo Mynard , Iman Almarzouqi. (2006) *Investigating peer tutoring,* ELT Journal Volume 60/1; doi:10.1093/elt/cci077. Published by Oxford University Press.
Keys, P. and Roberts, M.,(1991) Information Systems Development and Soft Systems Thinking: towards and improved methodology. In *Systems Thinking in Europe,* Plenum, London.
Lai, L.S. (2000) An integration of systems science methods and object oriented analysis for determining organisational information requirements. *Systems Research and Behavioural Science* 17, 205-228.
Krutchen, P. (2000). *The Rational Unified Process – An Introduction*. 2nd Edition. Addison-Wesley.
Miles, 1992; Prior, 1990; CCTA, 1993; Stowell and West, 1994).
Miliszewska Iwona , Tan Grace (2007) *Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming*. Issues in Information Science and Information Technology, volume 4, 277-289.
Mingers, J., (1988) Comparing conceptual models and data flow diagrams. The Computer Journal 31 (4) 376-379.
Mingers, J., (2001) *Combining IS Research Methods: Towards a Pluralist Methodology*, Information Systems Research, 12, 3, Institute for Operations Research and the Management Sciences (INFORMS), pp. 240-259.

Miles, 1992; Prior, 1990; CCTA, 1993; Stowell and West, 1994).
Prior, R., (1992) Linking SSM and IS development. *Systemist* 14 (3) 128-132.

Pawson R. & Mathews R. (2002) *Naked Objects*, John Wiley and Sons Ltd, West Sussex, England.

Salahat et al. (2008) A systemic Framework for Business Process Modelling and Implementation, In the proceeding of 5th International Conference on Innovations of Information Technology (Innovations'08), UAE University, Al Ain, UAE, in IEEE xplore 978-1-4244-3397-1/08.

Sarah loos, et, al.(2006)*Three Perspectives on Peer Tutoring for CS1*, in the proceedings of the Midwest Celebration of Women in Computing conference.28-31.

Sewchurran, K. & Petkov D (2007) *A systemic Framework for Business Process Modelling Combining Soft Systems Methodology and UM*L, Information Resources Management Journal*,* 20, 3, IGI Publishing, PA,USA, P. 46-62.

Stamouli, I., Doyle, E., & Huggard, M. (2004). *Establishing structured support for programming students.* Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference, Savannah, GA.

Stowell, F. and West, D. (1994) *Client-Led Design*. McGraw-Hill.

Svatopluk Štolfa, Ivo Vondrák, (2006) *Mapping from Business Processes to Requirements Specification*, Retrieved on 7th Aug, 2008 from 85.255.195.219/conf/esm/esm2006/abstract.pdf

Wade, S. and Hopkins, J., (2002) *A Framework for Incorporating Systems Thinking into Object Oriented Design,* Seventh CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02), Toronto, Canada, 27-28.

Warboys, Brian, Kawalek, Peter, Robertson, Ian, and Greenwood, Mark. (1999) *Business Information Systems-A process approach*, McGraw-Hill, UK.