Application of a Systemic Soft Domain-Driven Design Framework

Mohammed Salahat, Steve Wade, Izhar Ul-Haq

***Abstract*—**This paper proposes a "soft systems" approach to domain-driven design of computer-based information systems. We propose a systemic framework combining techniques from Soft Systems Methodology (SSM), the Unified Modelling Language (UML), and an implementation pattern known as "Naked Objects". We have used this framework in action research projects that have involved the investigation and modelling of business processes using object-oriented domain models and the implementation of software systems based on those domain models. Within the proposed framework, Soft Systems Methodology (SSM) is used as a guiding methodology to explore the problem situation and to generate a ubiquitous language (soft language) which can be used as the basis for developing an object-oriented domain model. The domain model is further developed using techniques based on the UML and is implemented in software following the "Naked Objects" implementation pattern. We argue that there are advantages from combining and using techniques from different methodologies in this way.

The proposed systemic framework is overviewed and justified as multimethodologyusing Mingers multimethodology ideas.

This multimethodology approach is being evaluated through a series of action research projects based on real-world case studies. A Peer-Tutoring case study is presented here as a sample of the framework evaluation process

***Keywords*—**SSM, UML, Domain-Driven Design, Soft Domain-Driven Design, Naked Objects, Soft Language.

I. INTRODUCTION

THE failure of software support systems has been well documented over the years, and many of these failures have been attributed to poor business process modelling (Joseph Barjis, (2008)). The systems failed because the business process model developed did not adequately support the process of designing and implementing the software support system. 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 [2].

Mohammed Salahat is with the Informatics department, School of Computing and Engineering, University of Huddersfield, UK, as a part-time-PhD student, and Lecturer with Ajman University in UAE. (e-mail: m.salahat@hud.ac.uk & abac.hasan.m@ajman.ac.ae)

Steve Wade is with the Informatics department, School of Computing and Engineering, University of Huddersfield as a Senior Lecturer (e-mail: s.j.wade@hud.ac.uk)

Izhar Ul Haq is with New York Institute of Technology, Abu Dhabi Campus, UAE, as Associate Professor (e-mail: ihaq31@yahoo.com)

There is a need for a systematic approach for capturing the information required by business processes [1]. This suggests a need to bridge the gap between business process modelling, information systems modelling, and implementation. Our previous work [4, 5] proposed and evaluated a development "framework" to deal with soft and technical systems aspects with an emphasis on modelling workflow. The evaluation results guided us to modify the framework in a new direction in which the concept of "workflow" is less dominant. The new modified framework focuses on Domain-Driven Business Process Modelling (DDBPM) as an approach to modelling business processes in an object-oriented domain model. This approach is named SDDD (Soft Domain-Driven Design). SDDD combines Soft systems Methodology (SSM), the Unified Modelling Language (UML), and the "Naked Objects" implementation pattern. SDDD aims to investigate, analyze and model a business domain so that we can implement it as a software support system. SDDD is a multimethodology systemic framework consisting of four phases with guiding procedures to steer the developer between the various compromises that need to be made throughout the development process. Section 2 reviews related work. Section 3 introduces SDDD and explains the basic structure of the framework. Section 4 and 5 then discuss the framework in more detail as a multimethodology approach. Section 6 is a brief description of a practical case study in which the method has been applied. Section 7 presents a reflection on the framework and the learning process of applying it suggesting further research.

II. RELATED WORKS

Domain-Driven design is an approach that seeks to model the system processes as a domain model and develop a software support system based on it. The first step of the DDD approach is to develop a Ubiquitous Language which consists of different concepts, diagrams, and documents to facilitate the communications between the developers and domain experts. The Ubiquitous Language will be used to create the domain model by the developers and domain experts [6]. UML defines a number of diagrams that can be used to model the business process [7] but lacks the ability to explore the soft issues related to the problematic situation which can be handled using Soft System Methodology. SSM [8, 9, and 9] 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. Some re-

searchers have explored the relationship between SSM and object oriented analysis and design techniques in general [11] but less has been written about the application of these techniques in the context of the UML. UML is considered by DDD to model the business domain as a "Domain Model".

Recent works [12]-[13] consider the SSM conceptual model as a focal point for linking SSM and UML by mapping the activities of an SSM conceptual model into UML use-cases. Recent examples of this approach can be found in SWfM [7] and our previous works [4]-[5]. The SDDD framework guides the developer into creating a "Soft Language" which consists of the output of the SSM stage to deal with the soft aspects which are not handled explicitly by Domain Driven Design. The SSM Conceptual Primary task Model (CPTM) is used to map human activity to a UML use-case model using a new elaboration technique. Use-cases, as abstractions of business activities, are used to model the business process in a domain model using UML diagrams and based on the philosophy of DDD which employs the idea of "Knowledge Crunching" during the different stages. SDDD employs the same philosophy during its four stages as explained in later sections.

Other researchers have made use of various extensions to the UML. For example [3] employed a systemic framework combining SSM and UML extensions proposed by [14] to model the business process of a manufacturing factory. Their framework is based on Mingers Multimethodology ideas [15] but does not encompass the software implementation phase of development.

Our previous works [4]-[5] presented a systemic framework for business process modeling and implementation as a workflow system, that framework was described as a multimethodology based on Mingers Multimethodology [15] and it compassed the software implementation phases of development. This paper aims to present an updated framework for modeling the system business processes as a domain model and implementing it as a software support system. The SDDD framework combines SSM, UML techniques, and the Naked Objects implementation pattern. To the best of our knowledge, this combination has not been applied in an intervention before.

III. DOMAIN-DRIVEN BUSINESS PROCESS MODELLING

The organization business process must be well defined and modelled for the implementation. A business process can be defined as 'the transformation of something from one state to another state through partially coordinated agents, with the purpose of achieving certain goals that are derived from the responsibility of the process owner' [16]. There are many definitions of "business process". Most of these definitions are based on the idea of a business process as a deterministic system that receives inputs and transforms into outputs following a series of activities. For example [17] defines business processes as "''structured sets of activities designed to produce a specified output for a particular customer or market''.

Good information systems software will support the organization work by handling the internal business process and control all aspects affecting the execution of the process. The business process must be supported with good business process modelling and implementation techniques that can analyze, model, and implement the business process in a professional way to achieve the organizational goals [18].

Domain-Driven Design can be used to model the business process as a business domain model [6]. A Ubiquitous Language (UL) is generated first as a communication tool between different stakeholders and the domain model will be generated and implemented based on UL. We have adapted the idea of a UL into a "Soft Language" which incorporate certain artefacts of a SSM analysis into the model. An object-oriented domain model can be extracted from this Soft Language through a transition process which will be explained in the next section. We argue here that SSM helps the developer to gain a deep understanding of different stakeholders' perspectives which will need to be represented in the Soft Language.

UML diagrams are sufficient tools for requirement modelling to support business process modelling in an object-oriented domain model [19]. When it comes to implementing the system we have made use of the DDD implementation pattern (i.e. Naked Objects) to reflect the system interface directly from the domain model. Naked Objects is described as "an open-source java based framework designed to encourage the creation of business systems from business objects" [20]. The latest version makes use of the Microsoft .Net framework.

V. SOFT DOMAIN-DRIVEN DESIGN FOR BUSINESS PROCESS MODELLING AND IMPLEMENTATION

The proposed framework is based on research into multimethodology, which justifies combining methods for the same business intervention [15]. It is
a multi-method framework which intended to guide the developer through an investigation of a problematic situation. The purpose here is to insure that a comprehensive understanding is achieved in order to facilitate the modelling and implementation of the domain-driven business processes as a software support system. The framework is being developed through a series of "action research" case studies. Action research requires the participation of the researchers in the development process. Accordingly our case studies have involved development projects within our own school. Our first two case studies have focussed on the development of a peer-tutoring-system and a support system for the school's combined studies programme. The researchers are part of the school and they are participating in daily activities related to the case studies.

The proposed framework SSDDDF (Figure 1) is focused on modelling and implementing of the domain-driven business process as a software support system. SSM is used as a guiding and learning methodology with techniques including UML and implementation pattern (Naked Objects)) embedded

within it. The DDD philosophy is adapted to generate a SL instead of UL and it will be an input to the next stages. The implementation pattern is used after the generation of the final refined change report which is an input to the implementation process. The research can't be a discrete event but a process that has phases with activities to be performed; the research process consist of four generic phases [15]:

1- Appreciation of the problematic situation and understanding why the problem exists as experienced by the involved actors.

2- Analysis of the methods and the data produced during the appreciation stage to understand how and why they are generated.

3- Assessment of alternatives that may be improve the current situation to better than it is, it includes interpretation of the results.

3- Action includes reporting about the results in order to recommend changes for improving the situation.

Using this generic model, the proposed framework consists of four phases and each phase consists of a group of activities. In the next section, the framework explained in details and "Evaluating the problem using SSM" stage consists of three activities represented in three steps [3]. The three steps equate to the appreciation, analysis, and assessment steps of Mingers generic model. Domain model generation takes place using UML modelling techniques because SSM lacks to techniques for taking actions [3], and this is equivalent to action step in Mingers generic model. In our framework, domain modelling and implementation is equivalent to action step in Mingers generic model. So, the proposed framework satisfies the generic process of conducting an action research in the business intervention. SSDDDF represented in Figure1, Figure 2 represents the conceptualization of the framework, and Figure 3 represents the logical processes embedded in it.

VI. THE FRAMEWORK OVERVIEW

The proposed framework consists of four phases and each phase consists of a group of activities. The details of these phases are as follows:

A. .Pre-SSM Phase

This phase consists of the following activities:

1- Initial problem identification

The problem in a specific area will be determined initially before starting the process of investigation.

2- Stakeholder roles analysis

This step to clarify the roles of all parities involved in the problem investigation to avoid any conflicts and to facilitate further proceeding into the other steps.

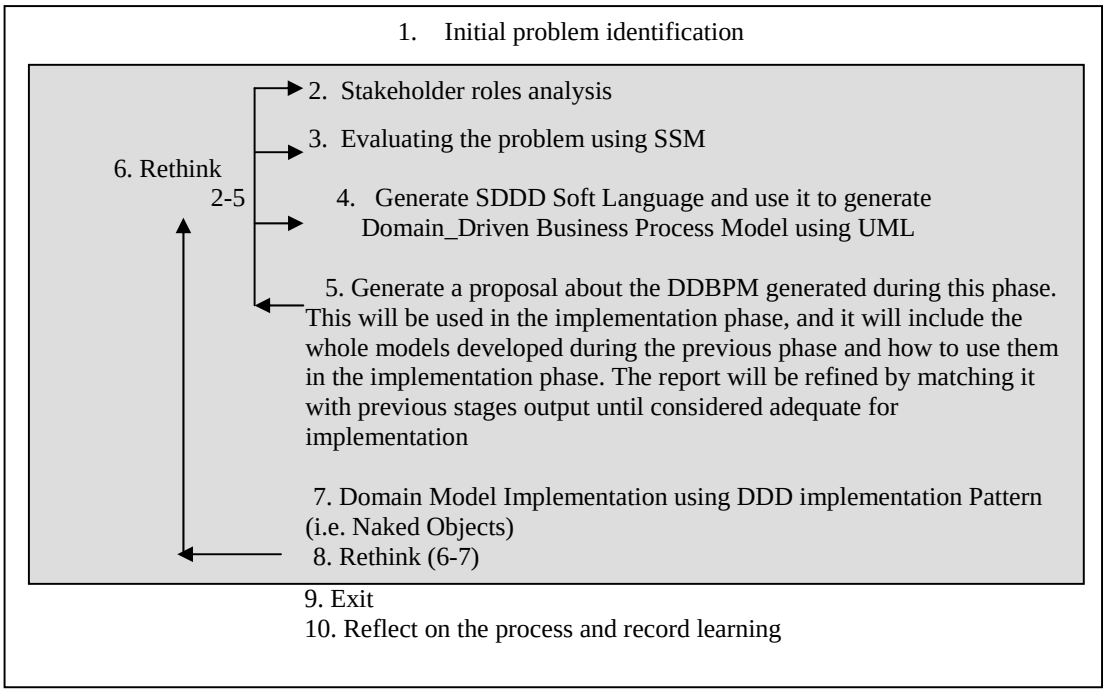

Fig. 1 A Systemic Soft Domain-Driven Design (SSDDDF)

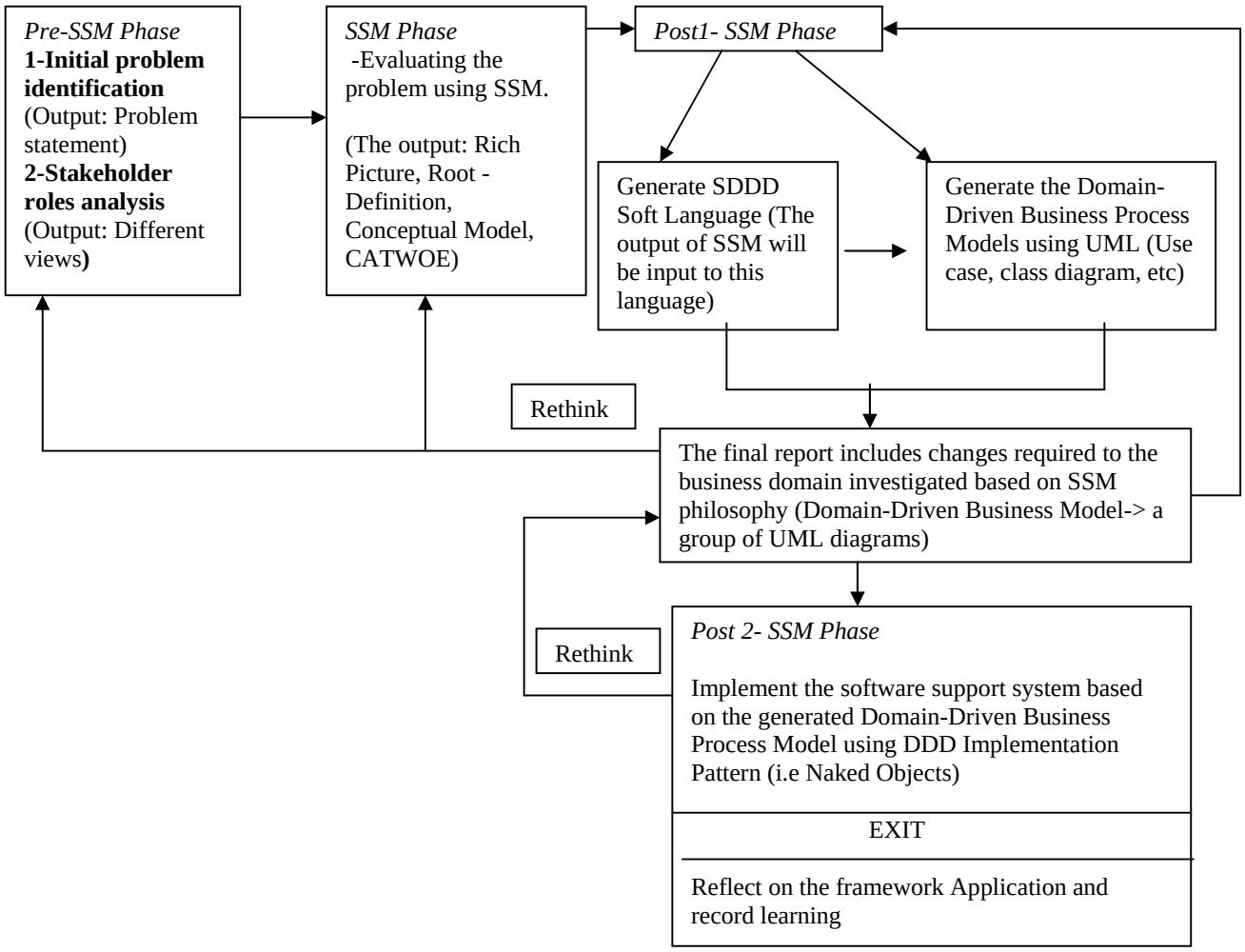

Fig. 2 The conceptualization of SSDDDF

b) SSM Application Phase

1- Evaluating the problem using SSM

SSM is a guiding methodology of the research and as shown in figure (1), there is a rethink about the steps (2-5) which includes the application of SSM to evaluate the problem; SDDD techniques are used to model the domain business processes and a change report will be generated which includes the modelled domain and how to implement it. The output of SSM stage will be an input to Soft Language of SDDD. This language is an important part of SDDD and represents the communication tool between the different stakeholders. SSM application consists of the following steps:

1.1 Investigating the problem situation using rich picture model

Anything can be included in rich picture and it is used to support the overall understanding of the organisation situation, goals, structure, and issues affecting the problem situation.

1.2 Modelling the relevant system using root definition and conceptual model

Root definition is used to determine the purpose of the system and the interested parties. Root definition constructs from the different views of parties concerned, and these views represent the expected functions of the system. Root definition represents the mission of the target system and look at the organization or the problem situation from different points of view. Root definition is one sentence and over all structure should be tested using CATWOE. (For details, see 8, 9, and 10). RD will be used to construct the conceptual model (CM) or consensus primary task model (CPTM) and it represents the human activity model.

1.3 Compare the (CM) with the real world

The conceptual model, as an abstract representation, will be compared to the real world (the current organizational process) for validation. If the organization business process model does not exist, then the conceptual model will be used as a basis to model it as a domain model [11]. The comparison will use the activities, organizational goals, objectives, and the structure using rich picture, root definition, and conceptual model.

2- SDDD Soft Language

Soft Language is the first product of SDDD. It consists of all documents and diagrams representing the business domain as communication tool between the different stakeholders. The proposed framework suggested that models developed following the Pre- Soft

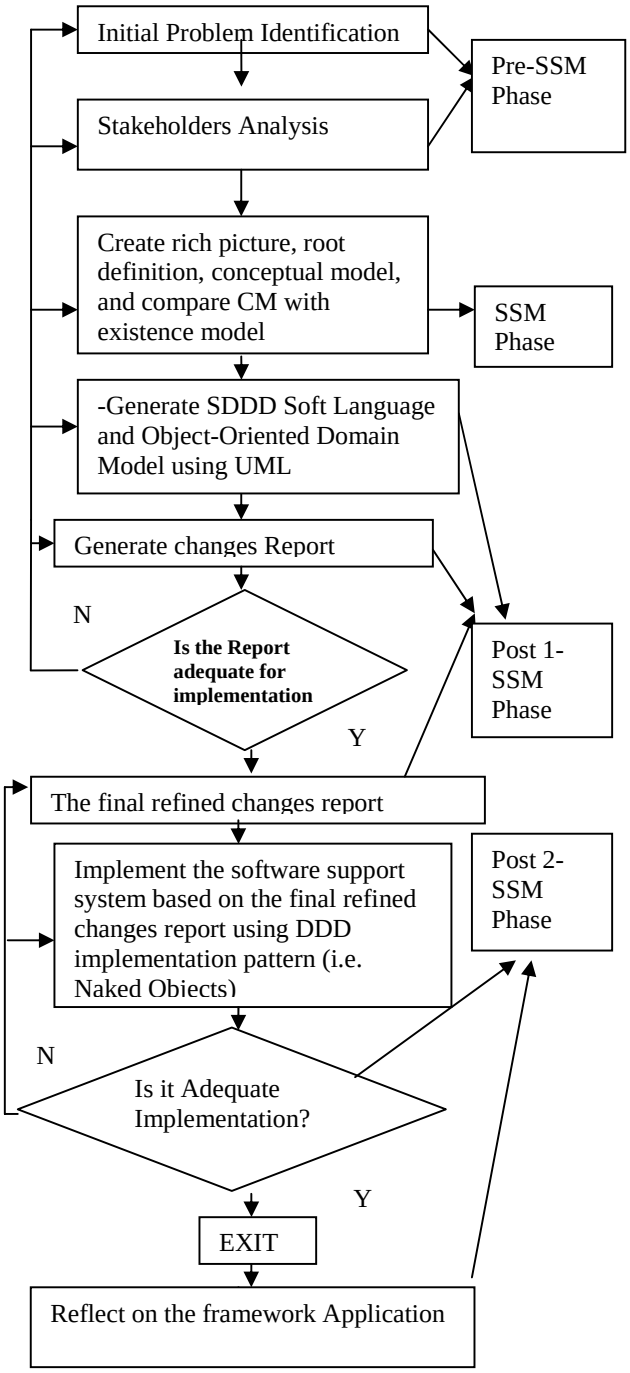

Fig. 3 The embedded logic in SSDDDF

Systems Methodology (SSM) and SSM Phases could provide useful input to the development of a soft language (SL). SSM helps the developer to gain a deep understanding of different stakeholders' perspectives which will need to be represented in the ubiquitous language (Soft Language).

C. Post1-SSM Phase

1- Object-Oriented Domain modelling using UML

The conceptual model (CM) or consensus primary task model (CPTM) is represents a general view of the domain

functional perspective. The decomposition of CM into subsystems will take place using a subsystem description table [11] and each subsystem activity will be represented in an activity description table. There is an identical similarity between conceptual model activities and use cases which make the conversion process possible and straightforward. A new elaborating technique is used to elaborate about any activity to be converted to a use case. This technique represented in Figure 6 and demonstrated through the case study.

1.1 Building a subsystem description and activity description tables

Subsystem description table will be prepared for each subsystem which includes subsystem number, name, head, and activities. Then, an activity description table will be prepared for each activity and it includes subsystem number and name, activity name, preceding and following activities, precondition, input and output data, tasks, business rules and constraints, post conditions, required skills and capabilities, role name, and performance criteria.

1.2 Converting the activities of the conceptual model into use cases

Activities will be tested to determine their goals, and some of the activities will be combined and some of them will be decomposed. The activities and their goals will be tested and mapped to UML use cases as one-to-one relationship. All use cases will be combined in the use case diagram which consists of use cases and their actors. The use case diagram is part of the use case model which is representing the organizational business process and it will be the basis for modelling the object-oriented domain model.

1.3 Use cases analysis and modelling

Each use case will be described using a textual format template. Each use case will be modelled using UML activity diagram, sequence diagram, and class diagram. The activity diagram is used to model the functional, informational, behavioural, and organizational work flow perspectives. The sequence diagram is used to model the interaction between the use case objects (the dynamic aspects of the workflow system). Finally, class diagrams for the static and organizational structures for each use case will be developed.

1.4 Developing the class diagrams

Class diagrams developed to model the behaviour of all use cases will be combined together in one class diagram called the analysis model. This model will be converted to a design model, by adding to it the design aspects required to design the object-oriented domain model.

2- Generate the changes a proposal

Change proposal to improve the domain model will be produced and it includes the whole models developed during the previous stages and guidelines for using them in the implementation stage.

3- Generate the final refined changes report

The report contents will be matched against previous stages results until an adequate report is a achieved.

D. Post2-SSM Phase

1- The domain model Implementation

DDD implementation pattern (i.e. Naked Objects) will be used in this stage because it's critical to start the implementation before refining the proposed modelling report. The domain model (mainly class diagrams) will be used to prototype the software system required.

2- Refining the implemented software support system

The implementation results will be matched to the refined modelling report and if any deviation available must be managed. This step represented in figure (1) as rethink (6-7).

3- Exit and reflect on the framework application

Exit implementation refinement step when an adequate software system reached. Then a reflection on the role of each component of the framework will take place. Finally, lessons learned from combining SSM, UML, and DDD implementation pattern will be recorded to guide further applications.

VI. THE CASE STUDY

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 implementation Pattern. 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.

A. Peer-Tutoring System Development

It aims to design and implement peer-tutoring system for introductory programming unit in the department of informatics to support the students and reduce number of failures. 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 [21], 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 [22]. Other researchers [23] 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. [24] 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 [23]-[24]. 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.

1- Pre-SSM Phase

1.1 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.

1.2 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.

2- SSM Phase

2.1 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.

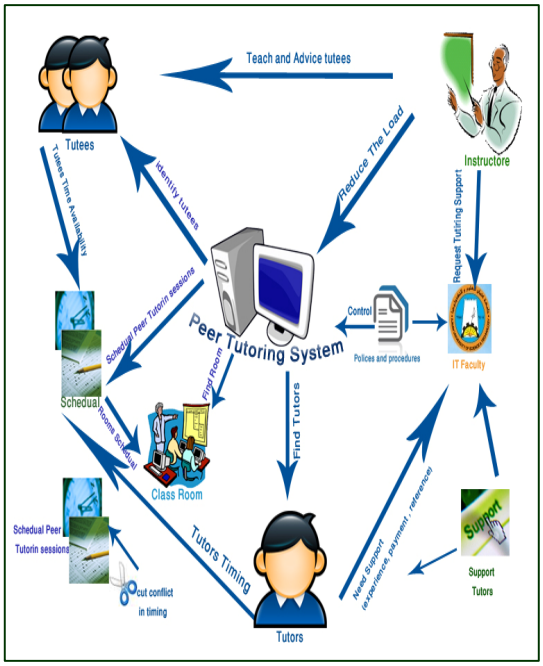

Fig. 4 Peer-Tutoring System Rich Picture

2.2 Modelling the relevant system using SSM

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 support. This presented in figure 5.

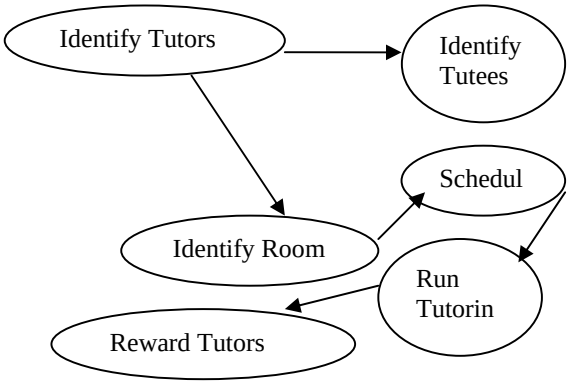

Fig. 5 CPTM of Peer-tutoring System

2.3 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 domain model. The CPTM, as a combination of all conceptual models, and by considering the other components of SL will be used in the next phase for to generate the domain model as stated in the beginning.

3- Post1- SSM Phase

3.1 Moving from SL to domain model using UML

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

3.2 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 (Fig. 6) shows the relationship between these key abstractions.

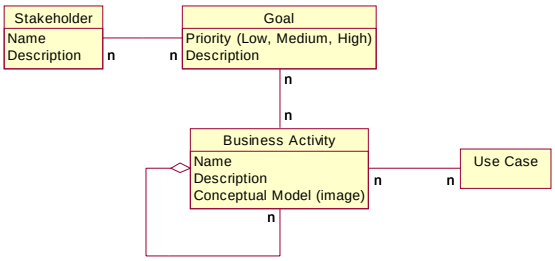

Fig. 6 Moving from an SSM to use case diagram

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 (Fig. 7).

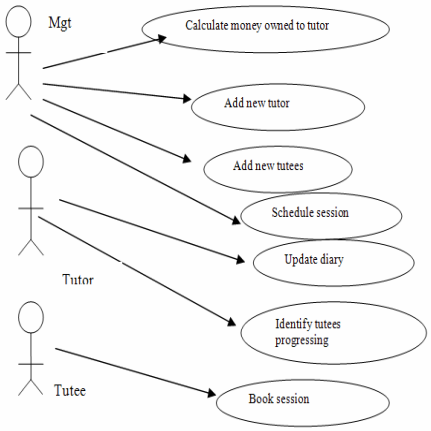

Fig. 7 Use case diagram

3.3 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. (Fig. 8)

3.4 Change report generation and refinement

As shown in the framework (SSDDDF), 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 domain model. As a guiding methodology, SSM focus on

the generation of the required change report as a result to be recommended for the management actions [8]-[9]-[10].

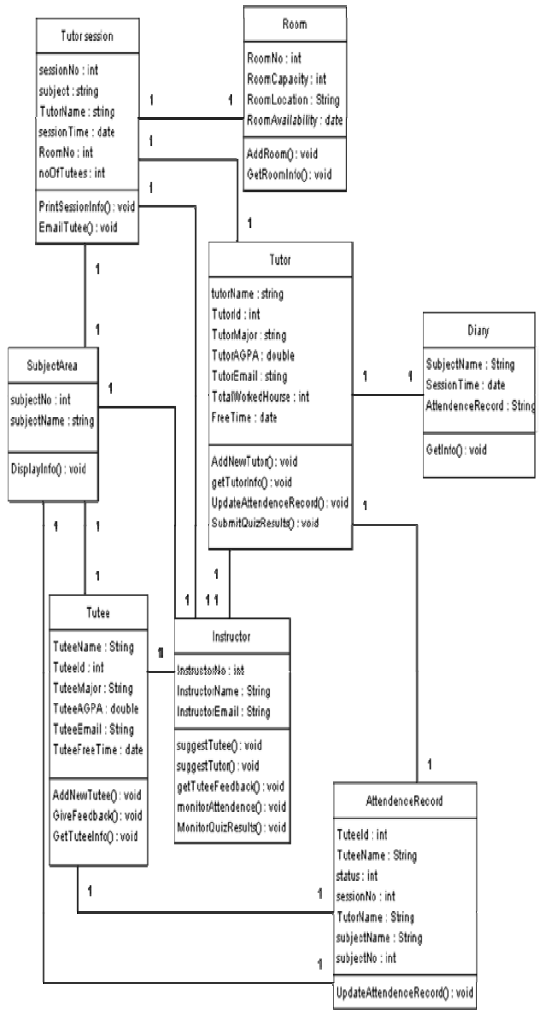

Fig. 8 Class Diagram of PTS

SSDDDF extended SSM further steps 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 domain 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 object-oriented domain model for implementation.

4- Post2-SSM Phase

4.1 Prototype Design, Implementation, Refinement

The class diagram is used to extract the domain objects which lead to a domain model which was implemented in VB and the Naked Objects implementation pattern. This process is 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: (Fig. 9)

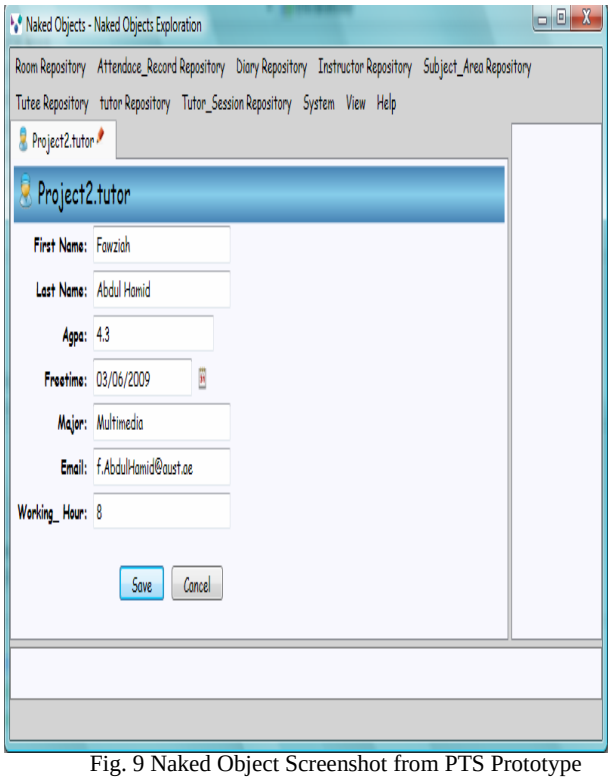

Fig. 9 Naked Object Screenshot from PTS Prototype

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.

B. Other case studies

This research is part of ongoing research aims to evaluate the proposed multimethodology framework using different case studies. Combined studies programs, work placement management systems, student associations systems, and others are a group of case studies which allow the action research approach to be applied by the researchers. We aim from this to find different important issues related to the framework in order to evolve as an ISD framework.

VII. REFLECTIONS ON THE FRAMEWORK

Our work in applying the framework to a series of real-world development projects has focussed our attention on a number of issues that we had not considered at the outset. Some of these present difficulties for the further development of the framework which present opportunities for further research. Some of these will be briefly discussed in this

section.

a) Role of Re-Use and Design Patterns in Domain Modelling

Our approach tries to preserve as much "soft" information as possible in the evolving domain models. Inevitably some of this information is lost as we move from approaches that try to model what "people" are doing (including activities that do not require software support) through to program code. At present our framework leads to development of a bespoke software system based on a rich object-oriented domain model. In practice many software developers make use of reusable software components or wish to design software with an eye to future reuse. There is clearly a tension between our emphasis on a bespoke solution and the software developers' objective of developing generic, reusable software solutions.

b) Representation of Implicit Information in the Domain Model

The conceptual models in SSM do not have rigorous syntax. We have discovered that when developing the conceptual models people often include information in, for example, the sequence of activities or the knowledge required to carry out certain activities which is lost when we move into the use case and object models. We are attempting to develop clear guidelines for identifying this type of information and what should be done about it. One possibility is that we develop our own version of conceptual models that do include a more prescriptive notation.

c) Ambiguity in the Definition of "Business Process"

One of the issues that we have confronted is the lack of consensus about precisely what can be defined as a business process and what cannot. SSM has a number of techniques for capturing multiple stakeholder perspectives on what the key business processes are and how they should be monitored. We want to preserve these multiple perspectives for as long as possible into the development process. At present we take the Consensus Primary Task Model produced in SSM to be an objective description of what is required but we have found that it is often difficult to gain consensus in developing this model and then to preserve that consensus as we move on.

VIII. Conclusion and Future work

The work done in this paper reviewed and highlighted the need for a multimethodology framework that can handle both soft and hard issues of domain business process modelling and implementation as a software support system. The new proposed framework is developed based on the idea of Domain-Driven Design (DDD) and Soft Systems Methodology (SSM). We have added a "soft" perspective on DDD to form "Soft Domain-Driven Design". The approach is described as a systemic framework for domain business process modelling and implementation. The framework is proposed and justified as a multimethodology framework, incorporating guiding steps through various key stages in the development process. The framework is being evaluated and further developed in an action research programme. We briefly provided the example of a "Peer-Tutoring-System" (PTS) case study to show how the proposed framework can be applied to a real problem situation. The evaluation work is ongoing in other cases including a "Combined Studies Programme Development" (CSPD) and the "Placement Unit Management System" within our institutions. More details will be the target of future publications.

References

[1] Joseph Barjis, "The importance of business process modelling in software systems design", Science of Computer Programming Journal, vol 71 ,pp 73–87, 2008.

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

[3] Sewchurran, K. & Petkov D, "A systemic Framework for Business Process Modelling Combining Soft Systems Methodology and UML", *Information Resources Mnagement Journal*, 20, 3, IGI Publishing, PA,USA, P. 46-62., 2007.

[4] Salahat , M., Wade, S., Lu, J., 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., 2008.

[5] Mohammed Salahat, Steve Wade. A Systems Thinking Approach to Domain-Driven Design. In the proceeding of UKAIS2009 conference, Oxford University, Oxford, UK, 2009.

[6] Eric Evan , *Domain-Driven Design –Tackling Complexity in the Heart of Software*, Addisson Wesley, 2004.

[7] Al Humaidan, F., *"Evaluation and Development Models for Business Processes"*, PhD thesis, University of Newcastle, UK, 2006

[8] Checkland, P., and Poulter J., "*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, 2006.

[9] Checkland, P., "Systems Thinking, Systems Practice", John Wiley and Sons Ltd, West Sussex, England, 1999.

[10] Checkland, P. and Holwell, S.E. , "Information, Systems and Information Systems, Making sense of the field", John Wiley and Sons Ltd, West Sussex, England, 1998.

[11] Bustard, D. W., Dobbin, T. J., and Carey, B. N., "Integrating Soft Systems and Object-Oriented Analysis", IEEE International Conference on Requirements Engineering, Colorado Springs, Colorado, pp. 52-59, 1996.

[12] Wade, S. and Hopkins, J., "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, May ,27-28,2002.

[13] Al-Humaidan, F., & Rossiter, N.," 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, 13-14 April,, 2004.

[14] Erikksonn, H. E., & Penker, M., "UML business process modelling at work", John Wiley and Sons, New York, 2000.

[15] John Mingers, "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, 2001.

[16] D. Platt, "Process Modelling and Process Support Environment to Design Management", Department of Civil Engineering, Faculty of Engineering, University of Bristol, UK, 1994.

[17] Daveport, T. h. Process innovation: Reengineering work through information technology, Harvard Business School Press, Boston, Mass, 1993.
[18] Warboys, Brian, Kawalek, Peter, Robertson, Ian, and Greenwood, Mark, "Business Information Systems-A process approach", McGraw-Hill, UK, 1999.
[19] Svatopluk Štolfa, Ivo Vondrák, "Mapping from Business Processes to Requirements Specification", Retrieved on 7th Aug, 2008 from 85.255.195.219/conf/esm/esm2006/abstract.pdf
[20] Pawson R. & Mathews R., "Naked Objects", John Wiley and Sons Ltd, West Sussex, England, 2002.
[21] Goodlad, S. and Hirst, B. *Peer Tutoring: A Guide to Learning by Teaching*,London: Kogan Page; New York: Nickols Publishing, 1989.
[22] Gardner, H. (1993) *Multiple intelligences: the theory in practice*. New York, NY:Basic Books.
[23] Miliszewska Iwona , Tan Grace. *Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming*. Issues in Information Science and Information Technology, volume 4, 277-289**.,** 2007.
[24] Hu Xiaohui. *Improving teaching in Computer Programming by adopting student-centred learning strategies, China papers, issue 6. 46-51*., 2006.