Pedagogical Evaluation of a Domain-Driven Design Framework

Mohammed Salahat
Ajman University of Science and Technology, UAE
m.salahat@ajman.ac.ae &
University of Huddersfield, UK,
u0423855@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 presents a pedagogical evaluation of the framework SDDD (Soft Domain Driven Design) which encourages a "soft systems" approach to Domain-Driven Design. The framework combines techniques from Soft Systems Methodology (SSM) with notation from the Unified Modeling Language (UML) and the "Naked Objects" implementation pattern. The framework has been used in the delivery of a postgraduate module in Information Systems Design for a number of years. This paper reports on the way in which the framework has been evaluated and improved during the teaching of this module and how it used for the development of postgraduate projects.

Keywords: Peer-Tutoring, SSM, UML, Multimethodology, Soft Domain-Driven Design, Modeling.

1.0 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 modeling (Barjis, J., 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 (Alter, S., 2007). There is a need for a systematic approach for capturing the information required by business processes (Barjis, J., 2008). This suggests a need to bridge the gap between business process modeling, information systems modeling, and implementation. Our previous work (Salahat et al, 2008), Salahat, M., Wade, S., 2009) proposed and evaluated a development "framework" to deal with soft and technical systems aspects with an emphasis on modeling 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 (Salahat et al, 2009), focuses on Domain-Driven Business Process Modeling (DDBPM) as an approach to modeling business processes in an object-oriented domain model. This approach was named SDDD (Soft Domain-Driven Design). SDDD aims to investigate, analyze and model a business domain so that it can be implemented in an object oriented programming language as a software support system. This paper focusses on an evaluation of the SDDD framework through teaching. Section 2 presents brief description of the framework. Section 3 reviews related work. Section 4 describes the way the module is taught. Section 5 describes the research methodology used. Section 6 describes changes made to the framework as a result of the evaluation.

2.0 Soft Domain Driven Design Framework

SDDD is a multimethodology design 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. The framework has been applied in a number of MSc development projects over the last five years. It has also been used as the "scaffolding" for a taught module in Information Systems Design. This module is based around the application of the Unified Modelling Language (UML) throughout

the development lifecycle from requirements analysis to implementation. All of the students arrive on the module with some background in modelling but students of the MSc Advanced Computer Science course tend to view modelling as high-level programming whereas those studying for MSc Information Systems Management tend to think in terms of business models. This presents the challenge of moving students into a deeper understanding from different starting points and with different preconceptions about the nature of the subject.

The structure of the framework is summarized in Figure 1.

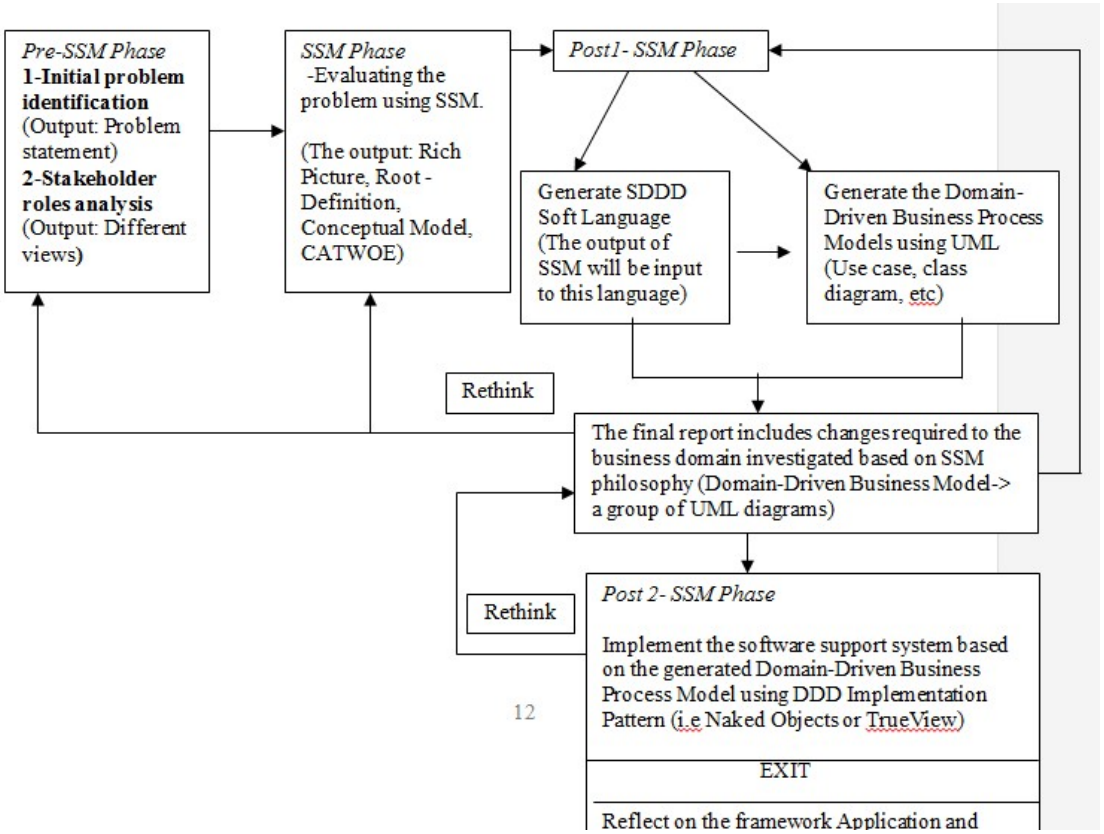

Figure 1 –The SDDD Framework

3.0 Related Work

3.1 Domain Driven Modeling (DDM)

The "Business Domain" comprises the business process that 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' (D., Platt,1994). There are many other definitions of "business process", and most of these 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 (Daveport, T., 1993) defines business processes as "structured sets of activities designed to produce a specified output for a particular customer or market''. The organizational business process, as part of the business domain, must be revised and modeled well and this required a proper modeling and implementation framework. To support the business domain, good information systems software used to 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 modeling (domain modeling) and implementation techniques that can analyze, model, and implement the business process in a professional way to achieve the organizational goals (Warboys et al, 1999).

3.2 Domain-Driven Design

Domain-Driven Design can be used to model the business process as a business domain model (Evan, Eric, 2004). 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 this UL. UML diagrams are sufficient tools for requirement modelling to represent the key objects and relationships in the UL represent the key objects and relationships in the UL and support the business process modelling in an object-oriented domain model (Svatopluk Štolfa, Ivo Vondrák, 2008). Within the SDDD framework we have made use of the Naked Objects design pattern to implement the domain model expressed in UML directly in software.

3.3 Soft Domain-Driven Design

Soft Domain Driven Design (Salahat et al, 2009), is an approach that seeks to incorporate techniques from Soft Systems Methodology (SSM) into domain driven design in order to model and implement the business domain. UML, as a part of SDDD, defines a number of diagrams that can be used to model the business process (Al Humaidan, F.,2006) but lacks the ability to explore the soft issues related to the problematic situation which can be handled using SSM.

SSM ((Checkland, P., Poulter, J., 2006), (Checkland, P., 1999), and Checkland, P., Howell, S.E,1998) is an established means of problem solving that focuses on the development of idealized models of relevant systems that can then be compared with real world counterparts. SSM is used in SDDD to model the business domain using rich pictures, root definition, and conceptual model. In our previous work (Salahat et al, 2009), we have adapted the idea of a Ubiquitous Language into a "Soft Language" which incorporate certain artifacts of a SSM analysis into the model.

The first step of the SDDD approach is to develop a 'Soft Language' as result of the application of SSM. This language compliments the Ubiquitous Language described in Domain-Driven Design (Eric Evan, 2004) which consists of different concepts, diagrams, and documents to facilitate the communications between the developers and domain experts. An object-oriented domain model can be extracted from this Soft Language via a transition process from SSM Conceptual Model to UML Use Cases. We argue here that SSM helps the developer to gain a deep understanding of different stakeholders' perspectives which will be needed to be represented in the Soft Language.

3.4 Other Related Works:

Some researchers have explored the relationship between SSM and object oriented analysis and design techniques in general (Bustard, D. W et al, 1996) but less have been written about the application of these techniques in the context of the UML. More recent works (Wade, S., Hopkins, 2002) and (Al Humaidan, F., Rossiter, N., 2004) 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 (Al Humaidan, F., 2006) and our previous works (Salahat et al, 2008), (Salahat, M., Wade, S., 2009), and Salahat, et al, 2009). Other researchers have made use of various extensions to the UML. For example (Sewchurran, K, Petkov, D., 2008) employed a systemic framework combining SSM and UML extensions proposed by (Erikksonn, H. E., & Penker, M., 2000) to model the business process of a manufacturing factory. Their framework is based on Mingers Multimethodology ideas (John Mingers, 2001) but does not encompass the software implementation phase of development.

4.0 Teaching with SDDD

We have followed the basic structure of the SDDD framework to develop an Information Systems Design module based around the following topics:

- How to use soft systems methodology to learn about a problem situation?
- How to extract Use Cases from the soft systems models?
- How to develop sequence diagrams related to each Use Case?
- How to develop a domain model from the collection of sequence diagrams?

- How to convert the domain model into a class diagram and database design?
- How to implement the class diagram as an object oriented software system using the, “naked objects”, implementation pattern?

We have used the basic step-by-step approach suggested by these topics as the basis for the design of the syllabus for our Information Systems Design module. The module is structured around a number of cases studies illustrating the method leading up to a “real world” design project based around the needs of our own academic departments in Huddersfield and Ajman University of Science and Technology.

It is beyond the scope of this paper to discuss the details of each of these topics or present the case studies that have been developed but the following diagrams are intended to give some idea of the deliverables developed when applying the framework. The diagrams relate to the design of a Peer Tutoring System. Figure 2 shows a rich picture produced in a recent tutorial.

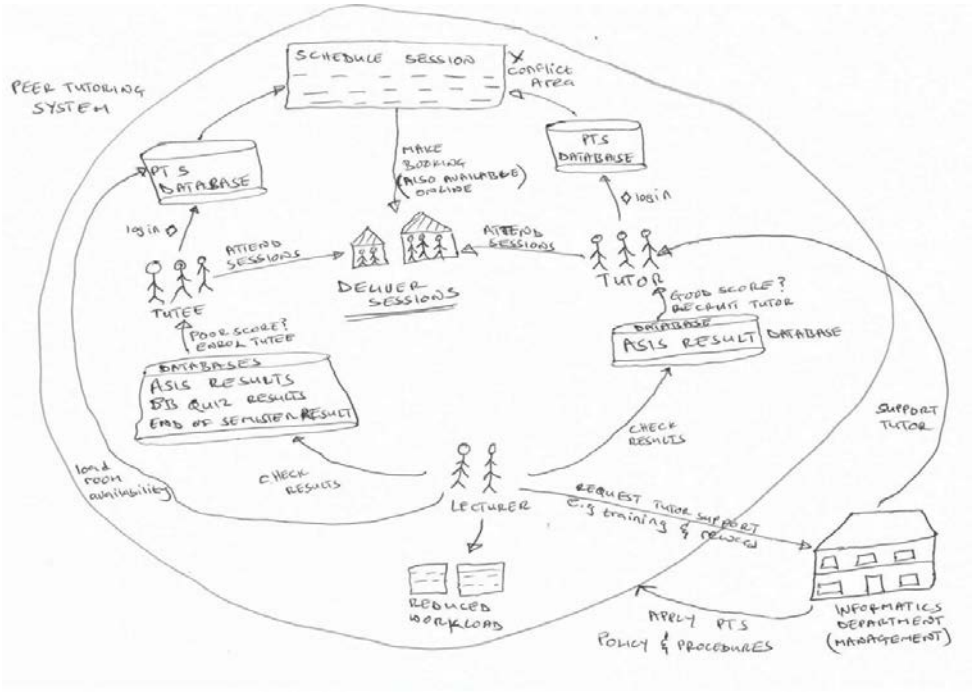

Figure 2 – An example rich picture

The discussion stimulated by the rich picture leads us into developing a root definition:

- A system owned by the school that provides study skills support to students using volunteers from the student body with the quality of their support activities monitored by academic staff.

Having agreed a root definition we move to developing more formal activity models such as the following:

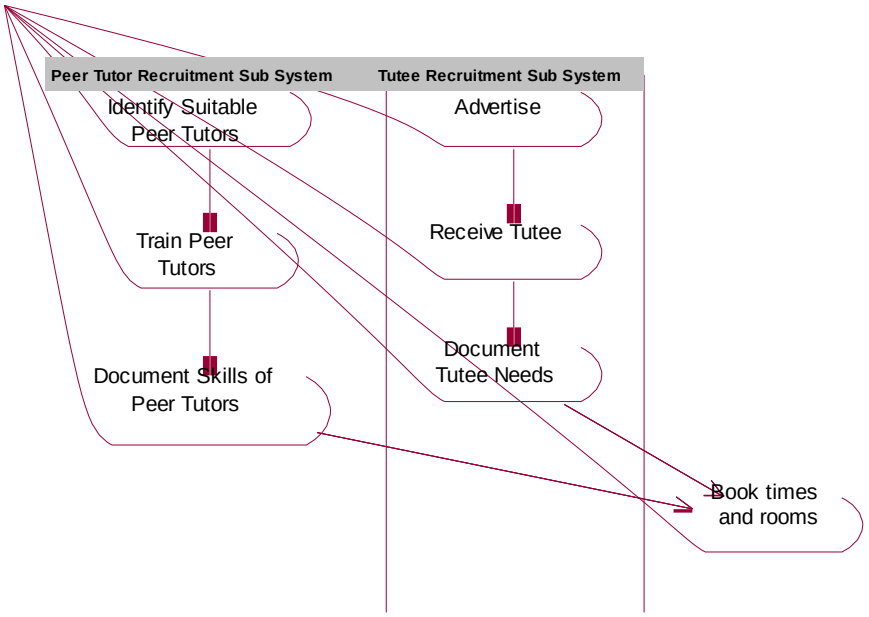

Figure3: Activity diagram of Peer Tutor and Tutee recruitment subsystems

These diagrams provoke further discussion. For example we might consider the following questions:

- Is it enough to advertise the peer tutoring service or should some (weaker) students be required to attend?
- Should we pay peer tutors?
- How should the effectiveness of the system be measured?
- Should we monitor the attendance of students at these sessions?

Some of these questions lead us to develop further activity models. For example Figure 4 for an "attendance monitoring" system.

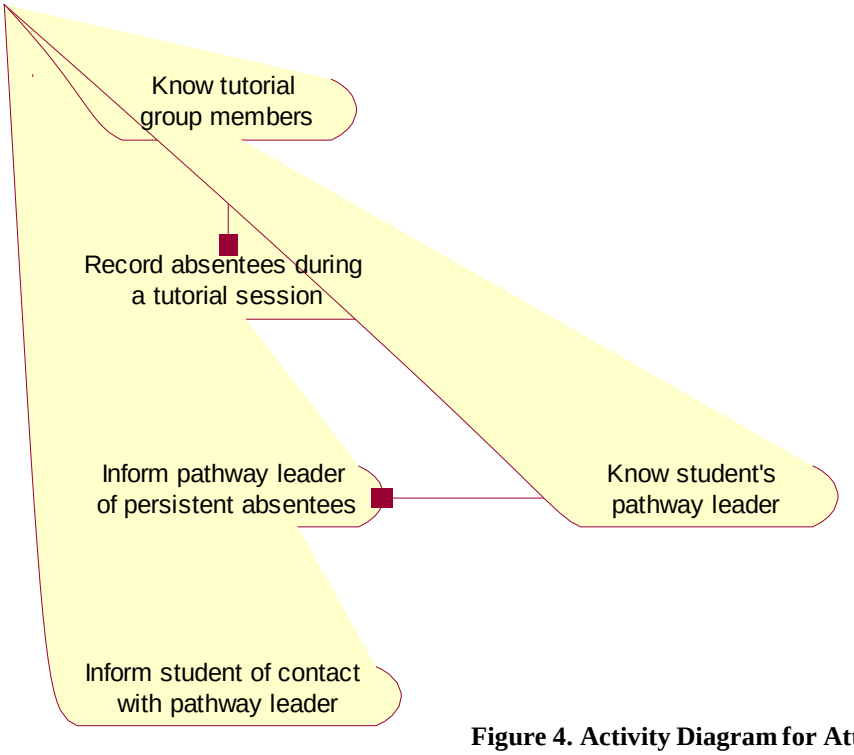

Figure 4. Activity Diagram for Attendance Monitoring

The following diagram could be derived from the activity diagram above:

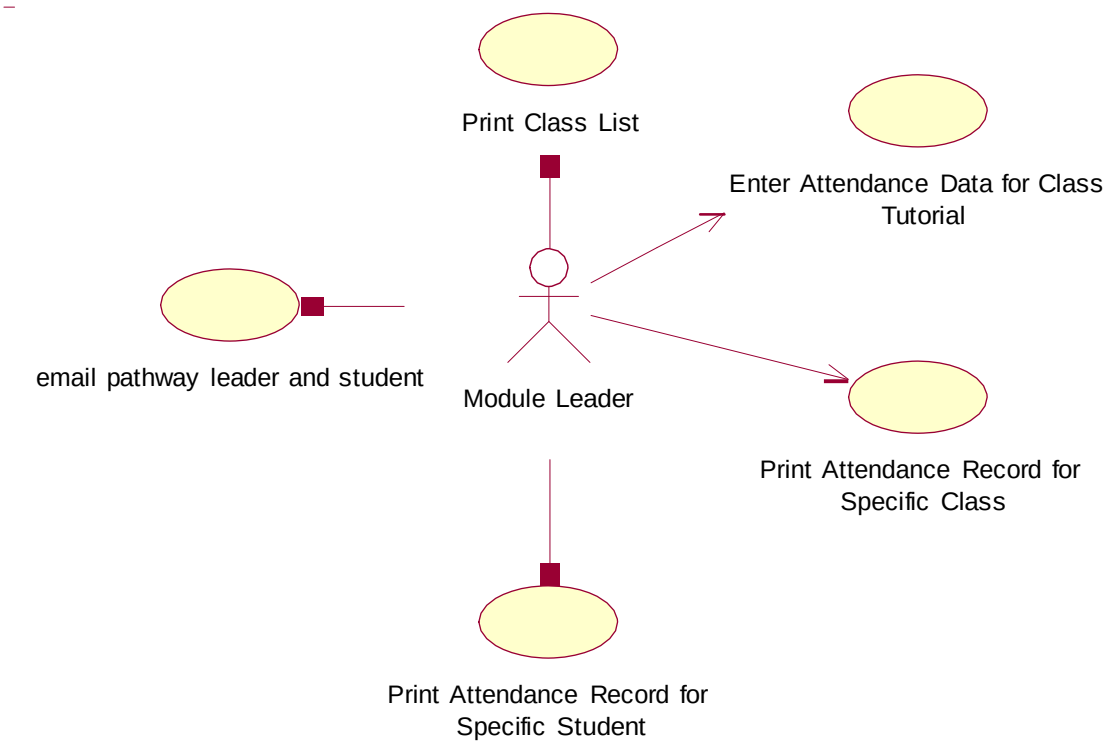

Figure 5. A Use Case Model.

If we focus on the "Print Class List" Use Case we might develop the following user interface in which the user enters a Module Code along with details of everyone who should be attending the peer tutorials for that module.

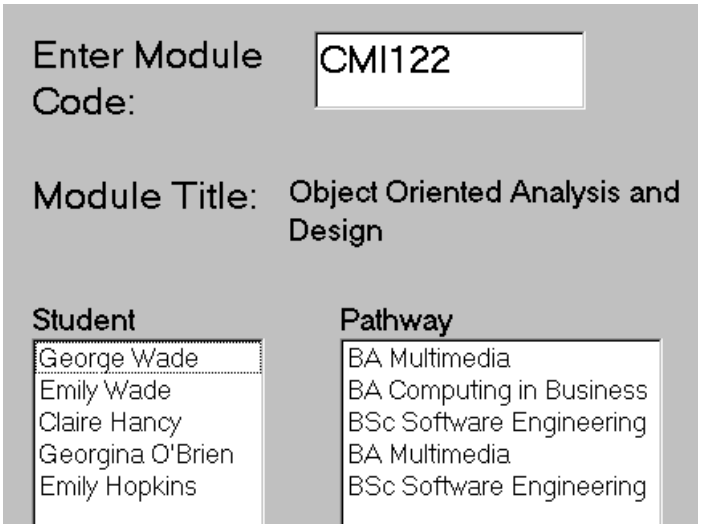

Figure 6. Screenshot for a Use Case.

To support this interface we could propose a sequence diagram explaining the role that a number of objects will have, "…behind the scenes".

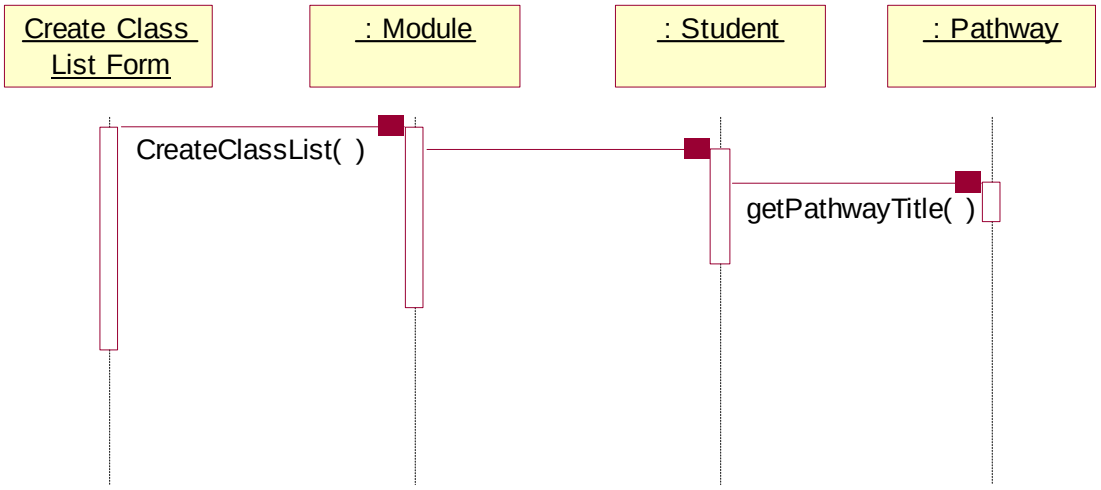

Figure 7. A sequence Diagram.

We would develop a diagram like this for every use case then develop a domain model consistent with all of these diagrams. A domain model derived from this one sequence diagram might look like the one presented in Figure 8.

We encourage students to use the Naked Objects Framework (2012) to generate object oriented code and a graphic user interface directly from the domain model. The interface generated in this way contains icons that represent each domain class in the model. In the above example I can select, "Module," by clicking its icon then search for a specific module and right-click on its "Create Class List" operation to see a list of students enrolled on the module. If I wish to enroll a specific student onto a specific module I can drag the icon representing that student on to the icon representing the module. Thus the relationship between the code and the model is explicit. From a teaching perspective this helps demonstrate that modeling is both about representing he real world and designing software.

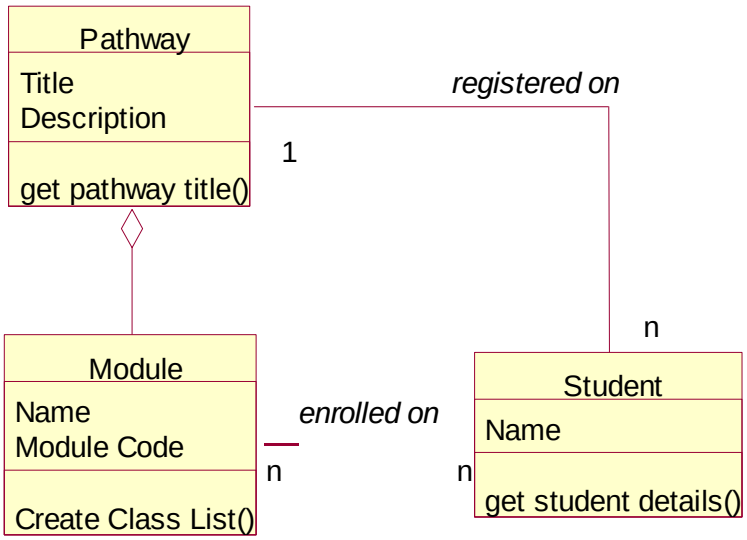

Figure 8: A Domain Model

5.0 Using SDDD in Development

5.1 Problem Identification

Salahat et al, 2009 indicated that 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 practice 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. The exact problem identified by working with the students as lecturers and interviewing them about the difficulties. The interviews were conducted with students studying programming modules in the Informatics Department at the University of Huddersfield as these will be the people using the system and students in the IT College in Ajman University in UAE. Feedback of authors located in both mentioned universities were recorded also. Also we interviewed some members of staff at the School of Computer Engineering as these are the people that will allow the system to be used in the department, reward the tutors and apply policy and regulations in the system. The stakeholders defined in this case are the people that will be using the system, and who will benefit from it. The stakeholders of the required PTS system were determined to be peer tutor, peer tutee, lecturer, and management. The stakeholders have different expectations of the system.

5.2 The SDDD Framework Application:

SDDD framework applied to build PTS application to help managing the system in the sense of scheduling, confirming and cancelling tutoring sessions for undergraduate programming modules. The application developed aims to help the administrator of the PTS in a way that it allows:

- Tutors book tutoring sessions without the help of a lecture, see how much rewards has been allocated to them and also update their diaries to allow tutees see if the tutors are available before making a booking.

- Tutees are able to book tutoring sessions without aid of a lecturer. They can also mark attendance for the sessions they attended to allow lectures and management judge progress of the system as a whole.
- Lecturers are able to load tutee, tutor and room information onto the system. The Lectures are also able to calculate rewards due to a tutor as per sessions they have delivered. Should there be a system failure; the lectures will also need to report them to an engineer to attend to the problem.
- The management is also to see the rewards allocated to a tutor by a lecturer so that they can be redeemed. Policies and Procedures will also be applied to the PTS by the management.

The system is developed and implemented using Naked Objects and TrueView implementation Patterns. The following are sample of models and screenshots from PTS project.

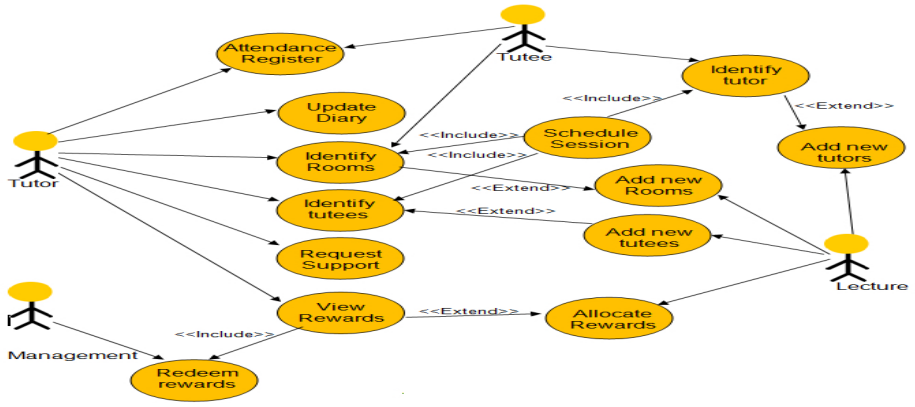

Figure 9: Peer-Tutoring System Use Case Diagram

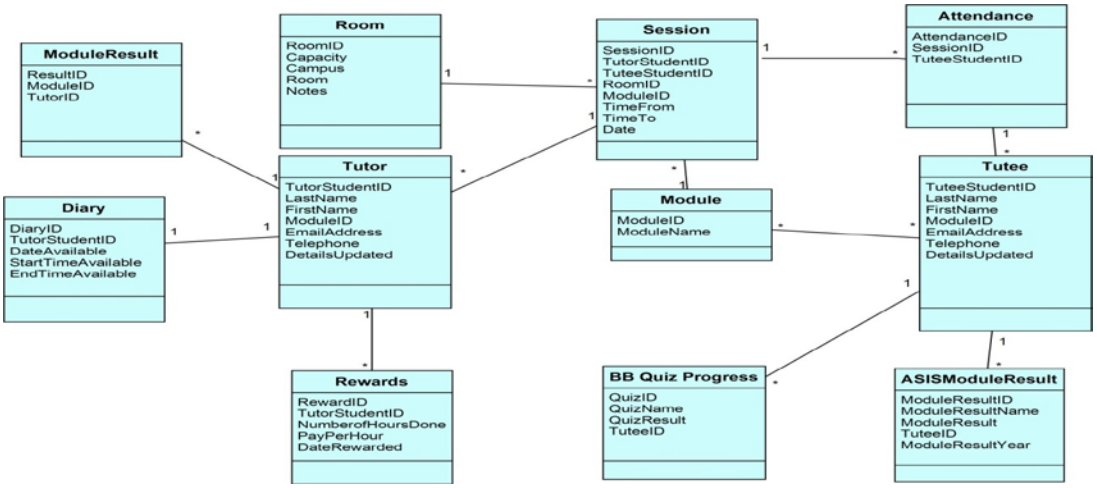

Figure 10: Class diagram

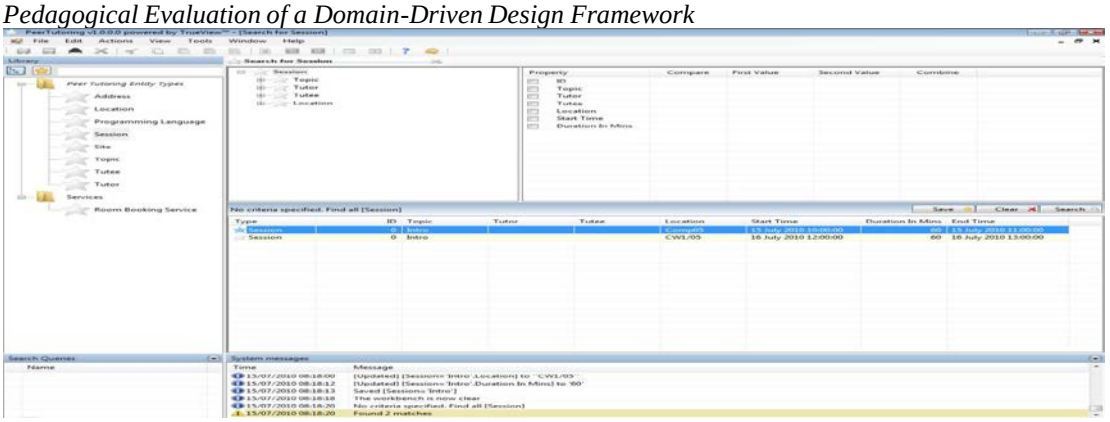

Figure 11: sessions booked in TrueView

Figure 12: Naked Objects MVC application with a user's mouse hovering over an object making Object behaviors directly accessible to the user.

6.0 Evaluation Methodology

The evaluation described in this paper was based around the use of the framework in teaching and development. In teaching, the following methods re used: a pre-course questionnaire, reflective essays, analysis of common mistakes in student work, in-class surveys and feedback questionnaire. Each of these will be discussed separately and the results of feedback questionnaire will be presented in another publication. In development, a postgraduate project is used to apply the SDDD framework and to reflect on its application.

6.1 Using SDDD in Teaching

6.1.1 Pre-Course Questionnaire

Thirty eight students joined the module in 2011. A background questionnaire was distributed to gather information about their prior learning in this area. It became apparent that two distinct types of student were studying the module:

- 18 students of MSc Advanced Computer Science. These students have a strong background in programming. Some experience of modelling but not with the UML. None of them were familiar with the idea of multimethodology. None of them had heard of SSM. Whilst studying methods and modelling these students are also studying advance software development modules in areas such as internet application development.

- 20 students of MSc Information Systems Management. These students do not have a strong back ground in programming. Most of them were unfamiliar with the principals of object oriented programming. Some experience of modelling but not with the UML. Most of them had heard of SSM but were not aware of the literature on multimethodology. Whilst studying methods and modelling these students are also studying information systems modules in areas such as competing in a digital economy.

6.1.2 Reflective essays

At the end of the module students were asked to write a reflective essay including a discussion on how the module reinforced (or otherwise) their appreciation of the techniques included in the framework.

These essays provided generally positive feedback. It could be argued that some students might have given positive comments in the mistaken belief that this might lead to higher marks. We did however make it clear to students that their objective evaluation was important to us as part of our action research project. The following are typical of the types of comment made in these essays:

- I did not know what modelling was or how it related to programming. I would like to apply these techniques on a real project.
- I know how to design systems properly now.
- I think this is the most important module because it links everything together.

Certain generalisations about the two groups can be made:

- Students of MSc Advanced Computer Science students seemed to regard modelling as high-level programming and were comfortable with the abstraction involved in developing class diagrams and sequence diagrams.
- MSc Information Systems Management students tended to see sequence diagrams and class diagrams as business models and to map them more directly to the real world.

This year we are presenting the modules to mixed groups so that each student will get to work with students on a different course.

6.1.3 Analysis of the common mistakes in the class work

A list of common errors would include the following:

- A lack of consistency between the SSM activity models and the Use Case Model.
- Operations required by the sequence diagram that are not included in the class diagram.
- Operations not supported by relationships or attributes.
- Database concepts (primary and foreign keys) used in the domain model.

We are working on developing the framework in ways that will steer future students away from these types of mistake.

6.1.4 In-class surveys

We used in-class surveys to evaluate understanding on a week-by-week basis. Typical comments included:

- I like way the framework gets you to specify a step-by-step approach. It made me think about what had to be done and why.
- It helps to organise the work in a framework. Each technique makes sense and they all work together.

6.1.5 Feedback Questionnaire:

The analysis still going on and the results will be presented in the next publication since it may be leads to detailed discussion and more stories which required more space to handle it them.

6.2 Using the framework in development

6.2.1 The Developer Feedback

The postgraduate student (the developer) provided the following evaluation and comments about the framework:

He mentioned that he has not come across any combination as this, the closest one he has come across is the one used by (Lane and Galvin ,1999) where they combined and transited from SSM to Object Oriented Analysis. In this they moved from SSM use cases and developed use cases but did not proceed to build an application using DDD implementation software, while in SDDD framework, the application is built allowing users to access the business objects without using controllers which Lane and Galvin did

not done that.

Also, SDDDF has so many advantages but the main one is that it enables and equips the researcher to understand the problem situation better through SSM as it tends to get different views of the situation from different stakeholders at the root definition stage and as well as at DDD stage when you have to understand the business objectives and how activities are done. This enables one to build a better application that would suit the user requirements and also build a system that has better requirements as studies in the UML stage. The application is even easier to use as it gives a user direct access to business objects and can manipulate them easier than through controllers like in conventional MVC applications.

He mentioned that he found the most difficult part of the framework is the conversion from SSM to UML, as this is not a one to one conversion, but it involves combination and decomposition of Conceptual models. He advised to do more research on this area to get a smoother and easier transition to ensure other researchers don't spend more time on I as he did.

About the implementation pattern he preferred Naked Objects rather than TrueView. The important issue he raised is the usability of the system developed using Naked Objects is better than the one with TrueView.

6.2.2 Reflections on the application:

We belief that this IS development framework it promotes is likely to be of interest to the software engineering community and, in particular, those who are also involved in teaching roles; its research method, arguments and recommendations are all in the context of that field. More practice in real projects in industry is required but it's not easy to approach them which encouraged us to depend on our postgraduate students projects. This makes an integration view between teaching and application of the framework in the same industry. This make more understandable for the students and us and this mad it more clearly to know how to use and practice the framework on both directions.

7.0 Conclusion

7.1 Generalization

This paper has reviewed our experience of delivering an Information Systems Development module following a development framework incorporating techniques

from SSM, the UML and the Naked Objects implementation pattern, and this framework applied in a real postgraduate student projects. The framework has been used as scaffolding to support the detailed content of the module. The resulting module structure has been fleshed out with a number of feedback mechanisms (i.e. in-class surveys, feedback questionnaires, focus group discussions and reflective essays) which have helped to finesse the structure of the framework. Feedback about using the framework in development presented and supported the refinement of the framework to be more applicable in practice.

7.2 Recommendations

We recommend more application of the framework either in academic or business industry. Different views will enrich the process of teaching using integrated framework like SDDD and the application development based on the same framework. Many frameworks and techniques available and well know but the potential results of using SDDD in both directions encouraged us to recommend it and we expected more used and succeed in the future.

We conclude that this approach has yielded a number of benefits and might have a wider applicability

REFERENCES

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

Al-Humaidan, F.,Rossiter, N.(2004) *Business Process Modeling 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.

Barjis Joseph (2008) *The importance of business process modeling in software systems design*, Science of Computer Programming Journal, vol 71 ,pp 73–87.

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

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

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

Checkland, P., Jim, S. (1990) *Soft Systems Methodology in Action*, Toronto: John Wiley and Sons.

Daveport, T., (1993) *Process innovation: Reengineering work through information technology*, Harvard Business School Press, Boston, Mass.

D. Platt, (1994) *Process Modeling and Process Support Environment to Design Management*, Department of Civil Engineering, Faculty of Engineering, University of Bristol, UK.

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

Erikksonn, H. E., Penker, M.(2000) *UML business process modeling at work*, John Wiley and Sons, New York, 2000.

Gardner, H. (1993) *Multiple intelligences: the theory in practice*, New York, NY:Basic Books.

Goodlad, S., Hirst, B.(1989) *Peer Tutoring: A Guide to Learning by Teaching*, London: Kogan Page; New York: Nickols Publishing.

Hu Xiaohui.(2006) *Improving teaching in Computer Programming by adopting student-centred learning strategies*, China papers, issue 6. 46-51.

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.

Lane, C., 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 pp12-13.

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.

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.

Naked Objects (2010) *Naked Objects MVC - Product description* [online] available at: <http://nakedobjects.net/product/product_intro.shtml> [accessed on 15th August 2010].

Oliver I., Kent, S. (2009) *Validation of Object Oriented Models using Animation*, [online] available at: <http://kar.kent.ac.uk/21768/1/validation_of_object-oriented_oliver.pdf> accessed on 24th June 2010.

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

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

Salahat , M., Wade, S., Lu, J.(2008) *A systemic Framework for Business Process Modeling 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..

Salahat, M., Wade S.(2009) *A Systems Thinking Approach to Domain-Driven Design*, In the proceeding of UKAIS2009 conference, Oxford University, Oxford, UK.

Salahat , M., Wade, S., Ul-Haq, I.(2009) *The Application of A systemic Soft Domain Driven Design Framework*, WASET online Journal, Issue57,pp476-486.

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

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., 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, May ,27-28.

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

Williams B. (2005) *Soft Systems Methodology* [online] available at: <http://users.actrix.co.nz/bobwill/ssm.pdf> [accessed on 18th June 2010].