A Reflective Approach to Improve Learning and Teaching of Software Engineering in Large Groups

Mohammed Odeh
Faculty of Computing, University of the West of England, UK

Abstract: *This research reports on the synthesis of a reflective approach to improve teaching and learning of software engineering in large groups. In addition, observations on the outcomes obtained from examination, group-project coursework, and informal feedback from students and tutors have been analysed. This resulted in discovering areas of deficiencies in undertaking group-projects, common areas where students' performance was similar in both exam and coursework. This study led to devise specific controls on the management of group projects and the implementation of tighter links between lectures and both practical sessions and group-work.*

Keywords: *Software engineering education, reflective learning and teaching.*

Received September 4, 2002; accepted May 13, 2003

1. Introduction

Although significant effort is usually exerted in teaching at higher education institutions, the outcomes at the end may not be in-line with the effort exerted at the first place. In more precise terms, the quality of the learning outcomes does not compare favourably to the effort spent in teaching. Is it the students, lecturers, and/or environment? In this research I investigate my experience of teaching software engineering subject in large groups and assess the outcomes of this experience with the objective of implementing a reflective approach to improve learning and teaching of software engineering in large groups. First, I introduce the research method used supported by a suggested reflective spiral process. Then, I present observations on the outcomes obtained from examination, group-project coursework followed by discussions on the informal evaluation of feedback from students and tutors, management of group-projects, and an agenda for a reflective lecturer/tutor. Finally, a conclusion is presented to summarize the main outcomes of this research.

2. The Research Method

Before describing the method I used in conducting this research, I first present the software engineering module being investigated and its aims. The Software Engineering module (UQC107S2) is taught at the University of the West of England (UWE), Bristol, UK. This is a second year module, for which I am the module leader that contributes to more than award but mainly to B.Sc. in Software Engineering, Computer Science, and Computing for Real-Time Systems. The intake in this module is around 191 students of mixed gender (168 male and 21 female), full and part-time students. There are 24 one-hour lectures spread over 24 weeks in one academic year. And, the practical sessions have the same schedule. I conduct the lectures, and there are three tutors including myself. In this module, a number of key-issues are stressed including team-work, the engineering discipline to software development [12] and software development process models.

In order to provide the skills and abilities required to develop software projects within a team working environment, students work on group-project coursework where they get exposed to project management, requirements engineering (where students gathered, analyzed, and specified requirements in addition to the use of system models), architectural design, user-interface design, and testing. In addition, each student submits an individual report that relates to the problems faced, lessons learned, and future enhancements of his/her undertaken project. Each group-work results were recorded as per attainments in the areas described above. In addition, the exam for this module was set in such a way to reflect on students' attainments in different areas as per the learning outcomes of the module. The exam results were carefully moderated and the marks obtained in each section of the exam have been recorded.

The data gathered above have been fed into worksheets to perform statistical analysis on the results in group-projects, project's individual reports, and exams. In addition, the results of students' attainment in the assignment and the exam have been compared, and particularly in similar subject areas, for example requirements analysis and software architecture. Also, students' and tutors' feedback have been analyzed.

3. The Reflective Process

The ultimate objective of this study is to use the results and conclusions obtained (based on the data

and facts gathered above) in a reflective manner, in order to improve learning and teaching of software engineering in large groups, and in particular at UWE. It is planned that this ultimate objective may be achieved by studying the following concerns:

- What went wrong in group-projects? Are there any areas of failure and how can they be classified?
- Were there any positive results? If there is any, what are they? Are they coursework-related, team-related, or both?
- Were there any special observations on running group-projects and project management skills?
- Were students at ease with using the software process model, modelling, and tools?
- Were there common areas where students' performance was similar in both exam and coursework? If there are any, could the level of students' attainment be related?
- Was the feedback gathered from students during group and practical sessions in-line with the outcomes of the above?

Once the outcomes of the above issues have been obtained, a case would be formulated with suggestions to enhance teaching and learning of software engineering.

A reflective process model has been used by adapting Boehm' spiral model [1] of software development, as shown in Figure 1. As a result, each loop of the spiral has been assigned the following four activities:

- Objective setting.
- Analysis of evidence gathered as shown in Figure 2.
- Implementation.
- Review

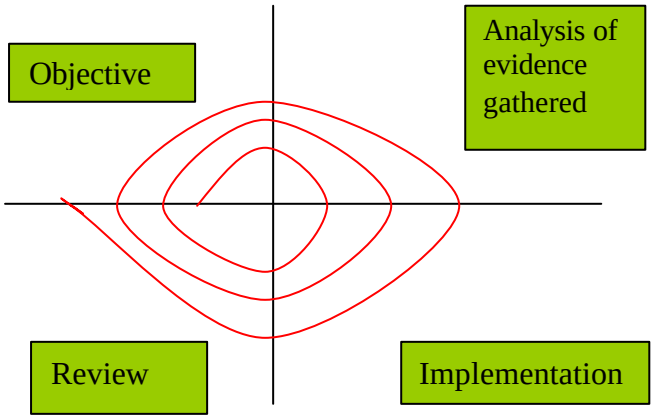

Figure 1. Adapted Spiral Model [1] for a reflective approach to learning and teaching.

It was reassuring to find a similar process suggested by Kurt Lewin who came up with the phrase "action research" in 1944, where the research described in this paper may be considered as an instance of action research [6]. Furthermore, the following quote from [6] highlights the resemblance between the process used here and the one proposed by Lewin: *"Lewin documented the effects of group decision in facilitating and sustaining changes in social conduct, and emphasized the value of involving participants in every phase of the action research process (planning, acting, observing, and reflecting)".*

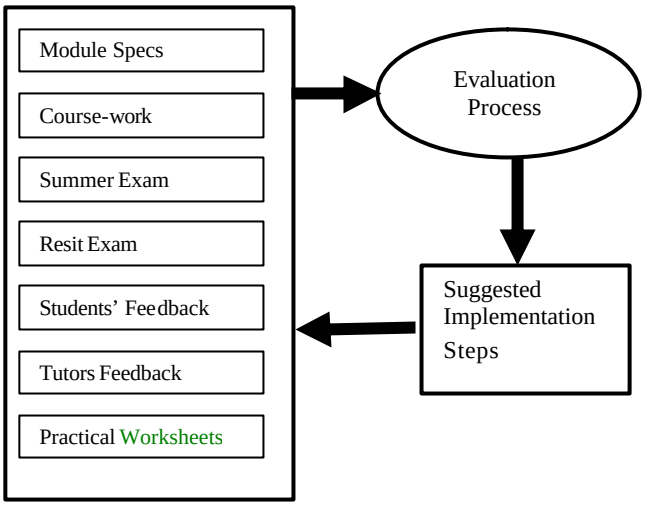

Figure 2. Input to, process, and output of the evaluation process.

This research represents the first loop in this reflective spiral process based on objectives set for this loop, analysis of evidence gathered based on the outcomes of running the module in 1999/2000, suggested implementation in 2000/2001, and then a review is required based on assessment of the results of 2000/2001 in order to start a new loop and set objectives for 2001/2002. Therefore, setting objectives and starting the second loop (iteration) is the subject of an extended study of this research. However, it is anticipated that two loops (in addition to the first one) are needed to tune the anticipated outcomes in a reflective manner. In addition, assessment of the need for a further loop needs to be decided at the end of a current loop should further iterations be required.

4. Outcomes of Undertaking Coursework

The coursework in this module is a group-work project. Students are required to be in a group of four or five. For ease of managing groups by tutors, group members should be officially timetabled for the same practical session and they will have to be in the same group that they sign up with initially for the life-cycle of the project (Guidelines implemented in 2000/2001). Though these are the guidelines, but experience tells us that managing groups of varying capability, attitude, promptness, etc. is not an easy task and never without problems. Group leadership is rotational as per major milestone or as students prefer it to be. The individual mark of each member in the group is based on group's results, his/her contribution to the group, and self- critical appraisal of the his/her experience in undertaking the project where students critically analyse the major problems faced in the project, lessons learned, and future enhancements to the run of the project as well as enhancements to the problem being tackled.

In order to be in-line with the key objectives of this module, students used a software development process in carrying out the different activities of the

assignment. In addition, they were required to use the Unified Modelling Language (UML) [2] to document the requirements and design models. Moreover, state-of-the-art tools were used to support the process they used, UML, and developing the system's prototype. As per group-work deliverables, students reported on the following:

- *Project management:* an initial project plan is submitted showing phases, deliverables, and tasks' dependency, if any. Changes to the initial project plan were required to be shown as per progress during the project in addition to the minutes of key meetings in the project. Finally, a Resource Allocation Table (RAT) showing who worked on which tasks and for how long. The RAT provides feedback on individual's contribution during the life of the project.
- *Requirements document:* with concentration on functional, non-functional requirements, system models, evolution of the system, and glossary of terms used in the project.
- *Design document:* the "4+1 views" [8] model of the software architecture are used to document the solution architecture with logical, implementation, process, deployment and use-case views. To further support creative thinking, students were encouraged to provide more views to describe the architecture of the system should they see the need for it.
- *Prototype:* students are required to develop Graphical User Interface (GUI) of the system being studied to show the key functionality of this system as per the requirements document above.
- *Software test specification document:* the test approaches and strategies are to be specified in addition to functional and non-functional requirements to be tested and included in this document.

Table 1 presents the outcomes of group-projects assessment with respect to project management, system models, requirements documents, architectural design, system prototype, and testing. Though students' performance in project management was not high, it may be considered acceptable given that this was related to second year students who were not involved in group project before except for some who (scored high marks) were part-time students and had the chance to work in teams before attempting this project. Furthermore, students had low performance in both system models and architectural design in particular with 36.64% on average. These two areas require practicing a number of new concepts and more creativity compared to other types of areas in the group-project. In order to improve on the performance of students in these two areas, one may think of enhancing the way practical sessions are run, increase the number of sessions in these areas, use techniques such as debates [13].

Table 1. Outcomes of group-projects coursework.

	Project Management	Requirements Document	System Models	Architectural Design	System Prototype	Testing
Average %	47.28 %	47.79 %	43.83 %	36.64 %	50.88 %	53.77 %
St. Dev.	22.21	31.39	15.3	26.13	13.02	26.68
Minimum	0 %	14.29 %	17 %	0 %	8.33 %	0 %
Maximum	90 %	75.71 %	75 %	79.17 %	90 %	90 %

Table 2. Outcomes of the exam assessment (Software Engineering Definition, Unified Process Model, Actors, Use-Cases, Non-Functional Requirements, Software Cost Estimation).

	SE-Def.	UPM	Actors	Use-Cases	NFR	SW Cost Estimation
Max	8	11	4	6	5	10
Min	0	0	0.5	0	0	0
Average	3.09	2.66	2.40	2.57	0.99	3.20
StDev	1.98	3.08	0.69	1.16	1.18	3.27
%Total Mark	38.63	24.16	59.90	42.86	19.75	32.00

Table 3. Outcomes of the exam assessment (Software Architecture, Object-Oriented Design, User Interface Design, Design Process, Configuration Management, Software Testing).

	SW Arch	OO Design Concepts	UI Design	Design Process	Configuration Management	SW Testing	Assign Related
Max	5.5	8	10	9	10	8	13
Min	0	0	0	0	0	0	1
Average	0.49	3.06	4.65	2.43	4.84	2.26	6.44
StDev	1.06	2.02	2.58	2.83	2.18	2.28	2.64
%Total _Mark	6.13	30.57	46.46	24.32	48.37	28.22	27.98

Table 4. Comparison of coursework to exam results.

	Requirements	SW Architecture	System Models & Design	User Interface	Testing
Coursework	47.79 %	36.64 %	43.83 %	50.88 %	53.77 %
Exam	34.44 %	6.13 %	27.45 %	46.46 %	28.22 %
Ratio Coursework/Exam	1.39	5.97	1.6	1.1 %	1.9

As a result, these proposals were implemented in the second run (2000/2001) of this module (except for debates which were done informally and not part of the assessment of the coursework) in the form of increasing the number of lectures and practical sessions in both subjects by extra two in each of these two areas. In addition, new worksheets were written to guide students on what is required as per specific milestones of the group project in addition to the instructions in the initial assignment distributed to students in the first term.

Students' performance in the other areas such as the identification and specification of requirements, system prototype, and testing ranged from 47.79 % to 53.77% on average. Though these results were below expectations, there was evidence that students had better results and confidence achieving the objectives of these areas compared to design-related one. This may lead us to say that students may have received better lectures and tutorials in these areas; furthermore, students were not faced with many concepts to practice compared to design-related ones. In addition, working on requirements is usually in the beginning of the project and working on the prototype and testing are towards the end of it. In these periods, attendance was observed higher in between these two periods compared to the design-related period. Thus, this may make low attendance an additional contributing factor to this low performance in general.

5. Analysis of Exam Results

There was one exam (a comprehensive one) that took place at the end of the academic year 1999/2000 and it constituted 60% of the final mark. The exam had compulsory sections and optional ones. The concentration in the exam was on assessing the learning outcomes in a number of areas including software development process models, requirements analysis, object-oriented analysis and design, software architecture, user-interface design, configuration management, software cost estimation, and testing. Tables 2 and 3 present the outcomes of assessing this exam in these areas. The data in these tables confirmed the same views obtained from analysing coursework results. Furthermore, there were other areas which were not part of the coursework and the students had variable performance, for example configuration management and software testing. It is worth mentioning that the low performance areas have been considered for improvements in the lectures and practical sessions with concentration on key issues (and less concentration on minor issues) that students need to learn at this particular level of the module.

In order to have a comparative analysis of the outcomes of both coursework and the exam, Table 4 was constructed to consolidate the figures in Tables 2 and 3 in the exam areas, which relate to the same areas examined in group-projects. Table 4 shows that the ratio of attainment in coursework to exam with respect to software architecture is the highest among other areas, which is an indication of very low attainment in this area. In addition, the figures in Table 4 confirm that students' performance in the coursework was better in all areas compared to the exam. This is attributed to the fact that coursework is a group-work and thus performance of students in the same areas in the exam varied as per individual's performance. I suggest calling the ratio of attainment in coursework to exam as the *average degree of variance in attainment between group-project (coursework) and exam.* Thus, this gives us an indication that although we encourage team-work as per the nature of software development in practice, there still remains the question whether this always leads to better attainment in the subject area on the individual level. On the on hand one may argue that this is the responsibility of the individual or the student. On the other hand, this necessitates the *need for the adoption of mechanisms to probe the individual attainment* within a group. One technique, which I found useful and used in subsequent teaching of this module in 2000/2001 was the use of meetings with groups as per scheduled milestones during the execution of group-projects. It would be valuable to see the feedback on this approach in a further evaluation of this approach in the following few months.

6. Discussion

6.1. Analysis and Assessment of Evidence Gathered

While running practical sessions during 1999-2000, it became very apparent to me that some students were not at ease when solving exercises in the pre-set worksheets, which were linked to previously taught lectures. Moreover, this was even more severe when those students were working on group-project related issues during practical sessions. A similar observation[1] was even found by Quintin Cutss while teaching first-year computer programming module at Glasgow University[2]:

"Observation of tutorial and laboratory performance indicated to me that many students had difficulty bridging the gap from relatively passive lecture material to active engagement in practical work, either on their own or at a machine." [3].

To understand the reasons behind this observation, I had to make informal discussions with students and tutors in addition to my own analysis, which led me to identify this as phenomena instead of an observation. I would like to call this phenomenon as "*low achievements in practical sessions*". I summarize

[1] In addition, I had a similar observation to this one while teaching a practical session in another module (Software Design using Java), which I did not lead.

[2] It is worth noting that both Cutts and I were experimenting this observation in different institutions and possibly overlapped in time.

below the main outcomes of studying this phenomenon:

1. Absenteeism: this was evident in practical sessions and lectures as confirmed from taking register in practical sessions and observations on the number of students in the lecture theatre. Students who confirmed their absence in lectures (in addition to their absence in some practical sessions) had variable reasons as to why they did not attend teaching sessions regularly. Some attributed this to working late shifts in jobs to support their study and living. Others had reasons such as illness, working on other assignments, commitments, placement interviews, simply "could not do it", etc. While this has impacted attainment on the individual level during practical sessions and the module as a whole, this had significantly (in a negative way) affected group work related to coursework. This is because attendance of group members undertaking a group project was vital to the coordination, communication, allocation of work, and achievement of pre-set milestones as per project plan.
2. Lack of follow-up between lecture and practical session: Though this may be considered as by product of absenteeism, there were still students who attended lectures and practical sessions regularly but had low performance in practical sessions. This was mainly attributed to the lack of follow-up by students between lectures and practical sessions.
3. Low commitment to achieve objectives of pre-set exercises in practical sessions: Unfortunately, this was evident in some students who showed lack of motivation. And, instead they were involved in other things during practical sessions such as replying to e-mail messages. Though tutors approached students to help raise their motivation and drag their attention back to the session, there were few cases whose behaviour were tolerated with difficulty.
4. Conflicting views between lecturer and tutors: This was evident while working on issues related to the group-project assignment as the software development process, the tools, modelling language were almost new in addition to being used the first time by tutors. This had led in some occasions to students getting conflicting feedback from the lecturer compared to what they had from their tutor in the practical session. The use of informal meetings for discussion between the lecturer and tutors, the updates on subject coverage in lectures using e-mail messages to tutors, and the provision of photo and electronic copies of material covered helped to minimize this conflict. In addition, I believe that this has been minimized to a greater extent now as this is the second time the module has been taught.

6.2. Project Management

Reading the individual reports of students, it was clear to me *that there was at least one problem* related to communication, coordination, team-leadership, missing project meetings, some members had no deliverables as per scheduled milestones, one or two persons dropping from the team, and/or one new person joining the team after few weeks have elapsed. These problems were of less impact in groups where students had better attitude towards working in a team, and the presence of mature students who had earlier experience in running projects in practice, for example part-time students. To help reduce the impact of these problems, the following were proposed for implementation in 2000/2001:

- Students were asked to submit an initial project proposal after 4 weeks from start of the project. A similar approach was found to be used in [4] and was useful in achieving a framework in the first few weeks of the project. This proposal represents a slim version of the vision document as per the inception phase of the Unified Process Model [7] which includes the team and its structure, description of the product features, actors, use-cases, project schedule, resource allocation table, resources required including hardware and software, and risks and risk minimization strategies. In addition, students were supposed to receive feedback from their tutors two weeks from the submission date.
- A number of worksheets were devised to guide students in running the project as per major milestones with a suggested duration span for each of these milestones.
- A number of pre-scheduled sign-off meetings between the tutor and each group were organised based on major deliverables. Although these milestones were announced ahead of tasks to be achieved, some very keen students suggested that this schedule should be incorporated as part of the coursework paper released initially. This had been taken into consideration for implementation in 2001/2002.
- The tutor participates in four key meetings during the life cycle of the project and he/she should sign off the minutes of these meetings in addition to signing off other four ones which he/she does not attend.
- Attendance of sessions to be recorded and any absent students to be contacted by their colleagues and informed of work progress so far. In addition, the faculty's administrator is to be informed of more than 3 consecutive (for 2001/2002 implementation).
- Although it was not formally announced to all students, students who were advised (or did it without being told) to create a shared repository of information about the project benefited a lot. This will be suggested to students and incorporated in the coursework script for 2001/2002.

Despite these suggested measures, I have to say that I agree with Quints's observation: "*There is an indication here that the attitude and previous experience of students will shape their ability to*

access new knowledge and their success on the course" [3] and found it a reality.

6.3. Agenda for Higher Education Lecturer

In general there is variety of methods in which lecturers can conduct their lectures or practical sessions, for example the utilization of students' interaction, good pace, presentation of an agenda of the lecture/practical, making a summary or recap of subjects covered at the end of the session [10]. But, the management of students' interaction is subject to group-size, subjects to be covered, and time management. In addition, there are characteristics (which are personal attributes) such as charisma and voice, which may not be easily changed, but they contribute to the achievement of an effective lecturer. Also, the lecturer should possess strong subject knowledge in order to make his/her lecture more effective, especially when practical and real-life examples are easily presented.

In addition to the above issues and in order to be more effective lecturer in higher education, one has to have the following as part of his/her internal agenda.

1. The lecturer has to *build coherence between learning and teaching processes*. This theme was supported by a number of theories from [11]. In a number of occasions we (as lecturers/tutors) place the blame on the student as why he/she did not follow instructions, achieve what was required in an assignment, exam, practical sessions, etc. Though there are some obvious occasions where some students take the blame (e.g. unexcused absences), but we need to recognize that teaching is part of the subject's knowledge and how it is to be learned by students as implied by Ramsden's 3rd theory of teaching "*Teaching as making learning possible*". For example, when I wrote the worksheets for practical sessions of the software engineering module, I planned to make what was conveyed in the lecture be well-understood and practiced in those practical sessions.
2. Lecturers needed to be facilitators and developers of critical thinkers so that we develop individuals who can develop themselves and the world around them [13]. In other words, we as lecturers need to be facilitators of critical thinking rather than doing critical thinking on behalf of students. The run of this module in 2000/2001 was improved to stress on team-work when attempting exercises in practical sessions followed by presentations and interactive discussion between the group presenting, other groups, and the lecturer. This shows that we have the soil watered (the infra-structure including the way the module has been organized, tutors, and tools) practically and we need to grow up the seeds (the individuals) right.
3. The lecturer needs to be a *reflective teacher* using his/her previous experience when he/she was a student and later becoming a lecturer.
4. The lecturer needs to be *adaptive teacher* in terms of being responsive to changes in the surrounding environments, emergence of new tools and techniques, and the special needs and diversity of students.
5. One needs to be aware of techniques/methods to enhance the learning process. For example, Kolb's experimental learning cycle: experience, reflection, conceptualization, and experiment. Although this happens in some incidents while teaching, but it is useful to highlight some examples in practice. For example, applying Kolb's cycle [9] to teaching and application of concepts studied in software engineering, one may build on the previous experience or background in undertaking a certain software engineering project, make a reflection on the software development experience, then conceptualize by developing new concepts, methods, notations, models, etc. Then, he/she needs to experiment these with respect to a software product to result in a new experience - which is full of concepts, models, notations, management experience, etc. - to apply when undertaking similar software engineering projects.

7. Conclusion

The ultimate objective of this study was to use the results and conclusions obtained from group projects in a reflective manner, in order to improve learning and teaching of software engineering in large groups, and in particular at UWE. This study has led me to devise more controls on the management of group projects. For example, students were asked to submit an initial project proposal four weeks after the assignment was released, with emphasis on initial understanding of the problem domain, team's structure, and initial project plan. In addition, tighter links between lectures and both practical sessions and group-work were implemented.

In general, I wish to have a tighter policy with respect to attendance in practical sessions (at least) as this had impacted group-work and attainment in practical sessions. Records of students' attendance were kept, but because there is no link between sessions attended and passing/failing the module, it makes controlling students attendance a tall objective to achieve.

The use of debates as a mechanism by which the skills of creative and critical thinking are developed is of paramount importance to the individual, society, and employers. Nevertheless, the great returns of this approach could be hampered by the high absence of students in practical sessions given the nature of working in a team and the activities involved in software engineering.

In addition, tutors and IT support staff need training on the use of tools. Also, tutors need extra time and budget to attend seminars presented by tool vendors in order to update themselves with new features provided by the tools for later enhancements to students' projects.

Team work and the development of team skills are vital not only for the accomplishment of the group coursework of the software engineering module but also for preparing software engineers who are ready to work within teams as it is the case in the industry. In addition, this will cultivate team spirits in individuals. Not only will this have impact on the individual, but also the society as a whole. Hence, this facilitates the achievement of one of the key objectives of educational research.

This research may be considered as generic model for other modules where attainment in exams, coursework, and feedback from lecturers, tutors, and students are studied in a reflective manner to improve on learning and teaching of a particular subject. Thus, this research could be classified as a generic one from which one can instantiate to create instances of reflective models to improve teaching and learning of different subjects in particular computing and other subjects in general.

Finally, I believe that this study is a step forward towards educational research that critically improves educational action as per Griffiths's definition "Educational research aims critically to inform educational judgements and decisions in order to improve educational action." [5]. In addition, the outcomes of the first loop of this reflective spiral process in 2000/2001 will evaluated further and the outcomes will be put forward for implementation in 2001/2002 in a reflective manner.

Acknowledgements

I would like to thank Prof. Melanie Walker for providing support and material related to educational research. Also, I am grateful to my colleagues Mrs. Chandriak Lakhani and Mr. Robert Storey-Day for their support in teaching practical sessions of the software engineering module in addition to their valuable feedback as tutors. Thanks are also extended to Dr. David Coward, Mr. Peter Maines, and Dr. Paul Raynor for their valuable support in implementing changes to the run of this module in 2000/2001. Last, but not least, I am grateful to King Fahd University of Petroleum and Minerals for the early experience in teaching this subject, and especially to Dr. JaraAllah AlGhamdi for the fruitful discussions about this type of research though it was in small groups.

References

[1] Boehm B., "A spiral model of software development and enhancement," *IEEE Computers*, vol. 21, no. 5, pp. 61-72, 1988.

[2] Booch G., Rumbaugh J., and Jacobson I., *The Unified Modeling Language*, Addison-Wesley, 1999.

[3] Cutts Q., "Fostering engagement, covering content: reflecting on large, mixed ability, first year computer programming module," in Walker M. (Ed), *Reconstructing Professionalism in University Teaching*, Open University Press, pp. 168-189, 2001.

[4] Favela J. and Pena-Mora F., "An Experience in Collaborative Software Engineering Education," *IEEE Software*, pp. 47-52, March/April 2001.

[5] Griffiths M., *Educational Research for Social Justice*, Open University Press, pp. 38-39, 1998.

[6] Kermmis S., "Action Research," in Hammersley M. (Ed), *Educational Research – Current Issues*, Open University Press, vol. 1, 1993.

[7] Krutchen P., *The Rational Unified Process Model: An Introduction*, 2nd ed, Addison-Wesley, 2000.

[8] Krutchen P., "The 4+1 View Model of Architecture," *IEEE Software*, vol. 12, no. 6, 1995.

[9] Kolb D. A., *Experiential Learning: Experience as the source of learning and development*, Prentice-Hall, New Jeresy, 1984.

[10] Newble D., and Cannon R., *A Handbook for Teachers in Universities and Colleges*, Kogan Page, 1991.

[11] Ramsden P., *Learning to Teach in Higher Education*, Routledge, London, 1992.

[12] Sommerville I., *Software Engineering*, 6th ed, Addison-Wesley, 2001.

[13] Warhurst C., "Developing Students' Critical Thinking: the Use of Debates," in Walker M. (Ed), *Reconstructing Professionalism in University Teaching*, Open University Press, 2001.

Mohammed Odeh is senior lecturer in software engineering and associate of the Complex Cooperative Systems Centre at the University of West of England, Bristol, UK. He holds PhD degree in computer science from the University of Bath, 1993 in addition to PGCert in Higher Education and membership of ACM and ILT. Dr. Odeh has more than 18 years of experience including extensive project management experience in planning and leading a range of IT-related projects in addition to management posts. He led the second work-package (the User-Requirements Specifications) of the MammoGrid project, an EU-funded project with collaboration from European partners such as Oxford University, Cambridge University Hospital, CERN, Udine University Hospital in Italy, and Mirada Solutions Limited. Dr. Odeh supervises five PhD students in bioinformatics, information management and integration, process modeling, and knowledge management. He also leads and teaches modules at both BSc and MSc levels in computer science and software engineering.