Int. J. Engng Ed. Vol. 24, No. 3, pp. 567–580, 2008
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2008 TEMPUS Publications.

A Methodology for Combining Development and Research in Teaching Undergraduate Software Engineering

HASSAN ARTAIL
American University of Beirut, Electrical and Computer Engineering, P. O. Box: 11-0236, Riad El-Solh, Beirut 1107 2020, Lebanon. E-mail: hartail@aub.edu.lb

In the most part, undergraduate students have been participating in research by working on faculty projects through which they primarily contribute to the implementation and testing of algorithms and systems. This paper presents a methodology that focuses on teaching students the skills of doing research. The approach taken is based on integrating a research component into a third-year undergraduate software engineering course. In this particular case, student groups studied a relatively large number of journal papers relating to a specific source software engineering topic in each semester the course was given, generated summaries, came up with ideas for research topics, pursued the research and wrote papers that described their work. We illustrate how the research component was integrated with the other components of the course, namely the software development project and lectures. The paper concludes with an assessment of what students have learned and a summary of the outcomes of the course in addition to the learned lessons.

Keywords: class projects; course design; student performance; teaching software engineering; undergraduate research.

INTRODUCTION

EARLY IN THEIR CAREER, professionals are likely to face situations that they did not encounter before and will discover that certain types of work can only be accomplished through collaboration with colleagues. In such situations, an employee has to rely on self-learning skills in order to better face and resolve the challenge. Research in its simplest form combines most of the skills required during the professional life and graduate studies. Although research is mostly stressed upon at the graduate level, some initiations at the undergraduate stage have proved to be fruitful.

This paper presents our experience in redesigning the Software Engineering (SE) course, which we have been teaching as a third-year undergraduate course. Undergraduate students take this course in their third year as part of the Computer and Communication Engineering program. The major change that was applied to the course is the inclusion of a research component that spans all the phases of a typical research activity. The goal was to expose students to research as part of the undergraduate program rather than expecting them to acquire the necessary skills by working on research projects of faculty members.

* Accepted 15 December 2006.

MOTIVATION AND BACKGROUND

Whether at the graduate or undergraduate levels, most research is conducted under the supervision of one or several faculty members. Involved students act as assistants and focus their efforts on the accomplishment of a certain project-oriented goal. In the majority of cases, the purpose of involving undergraduate students is to assist faculty in completing their research project objectives [1–3]. In such cases, the focus is set on using student's capacity to solve technical problems and write reports, rather than enhancing their knowledge and skills in doing research. Other goals for involving undergraduates in research have been cited but they were not presented as being the driving force, but rather as side benefits. These include preparing students for graduate studies [2] and applying for scholarships [4] and discovering student research interests [5]. Other reported benefits include increasing collaboration among students [2, 6] and between faculty and students [2, 5], making it possible for students to contribute to the design and development of systems and products [7] and involving underrepresented and minority groups in research [1, 2].

An indication of the attention that is being given to undergraduate research is the emergence of journals dedicated to the topic. Among those we mention the Reviews in Undergraduate Research [8] and the Caltech Undergraduate Research Journal [9]. Moreover, the importance of undergraduate research has been widely acknowledged by academia, industry and governmental agencies. The Council on Undergraduate Research was established to promote undergraduate student–faculty collaborative research and raise the awareness of governmental and private institutions on the importance of undergraduate research [10]. Similarly, the National Conferences on Undergraduate Research, which sponsors annual conferences, focuses on the promotion of undergraduate student achievements [11]. The US National Science Foundation (NSF) has even set up a program, called the Research Experiences for Undergraduates (REU) [12], to financially support and promote research participation by undergraduate students. The objective of this program is to seriously involve undergraduate students in research projects and activities that are specifically meant to foster research skills in these students. The REU Website explains that this program indirectly aims to attract undergraduates to and retain them in careers in science and engineering. Related to the theme of this paper, the NSF site attests that research experience is one of the most proven methods for interesting talented students in fields linked with exploration and academic research.

In the following, we present prior works that share similarities with our approach in that they deal with introducing students to research through SE class projects or with providing practical experience in developing software.

In an SE class, students were required to develop software systems by inferring requirements from selected technical papers for learning how to handle incomplete requirements and developing skills for gathering missing information from the literature [13]. A real-life situation was simulated by students taking on different roles (managers, analysts, developers, etc.). This approach did not offer students a complete experience, as it was limited to the initial stages of doing research. Second, by having group members take on distinct roles, students were not able to apply many of the learned methodologies in the project. Reportedly, this class provided a motivation for including few undergraduate students in a two-year funded research project, in which half of the students were paid from the grant while the others volunteered [14].

It was reported that volunteer students were not as effective and many left the project before its conclusion. An issue with depending on funded projects to expose undergraduates to research is that only a limited number of students tend to be selected based on academic performance and expressed interest. As a result, lesser performing but capable students are often left out. Second, funded projects usually require long-term commitment and may not be suitable to all students especially if they have to complete final-year projects and undergo practical training. In another reported experience, students engaged in SE practical training through two projects to practice SE methodologies [15]. They received detailed documentation with examples and were actively supervised. In addition, students were obliged to work a specific number of hours in the laboratory. With this approach however, students missed on the opportunity of autonomously investigating problems and developing solutions from scratch.

Many universities require their undergraduate students to complete a research project before graduating. An example is the Senior Research Project that is taken by Electrical and Computer Engineering undergraduate students at the University of Illinois at Urbana-Champaign (UIUC) [16]. The course includes individual research work, laboratory experiments, computer simulations and software development. At the end, students have to submit a written research proposal that includes the results of their research. This course has a continuation through a senior thesis, in which students have to complete the research and submit a thesis accompanied with an oral presentation.

Many faculty members who have written about the subject of undergraduate research have acknowledged the inadequate level of student educational background for conducting graduate-level research [17]. A while ago, a group of researchers had identified the desired characteristics of curricula reform in undergraduate education [4], with the broad categories being: 1) depth of knowledge in specific areas; 2) involvement in teamwork; 3) experience with open ended problems; and 4) thorough approach to problems and career. Taking the above into consideration, we designed the research part of the course to methodically teach students the individual steps involved in performing research and having them apply these steps toward conducting an actual research project of their own. The Software Engineering course, which we taught for the first time in the Electrical and Computer Engineering (ECE) department at the American University of Beirut (AUB) in the spring of 2004, was the right vehicle to achieve the above goal, given that the subject of Software Engineering is diverse and can be linked with the many research topics that interest students. Since then, we have also taught the course in the fall of 2005 and in the spring of 2006, while using the same course design.

COURSE DESIGN

Traditionally the course of Software Engineering had two components: a development project and two examinations. The project required students to work in groups of two or three on the design and implementation of a practical software solution. Students were usually asked to approach local organizations and offer them the opportunity to develop moderately complex software systems fully, free of charge. Traditionally, students were able, in the most part, to make such arrangements with small businesses in the service industry and these businesses had to agree to hold meetings with the students at their sites or at the university to provide the necessary documentation. Through this project, students get the opportunity to meet real customers and apply the software engineering methodologies that they have learned in the classroom, beginning with requirement gathering and ending with testing and deployment.

The major change we introduced to the course is the inclusion of a research component, which we describe in detail later in the paper. With this new component, we were still able to provide full coverage of the basic SE topics and to engage students in practical software development. We used the book by Pfleeger [18] and covered chapters 2 through 13, which cover topics that are normally included in a basic software engineering course. The semester included 32 1½-hour lectures, 22 of which covered material from the book, four discussed case studies and the balance were taken up by group presentations and project discussions. Students were expected to apply the SE methodology to the various aspects of the software development project. In particular, they were responsible for generating and submitting to the instructor requirement specifications, design documentation (including UML diagrams), design complexity reports and Gantt charts for reporting group and individual progress. Furthermore, at the end of the semester, students had to conduct a demonstration of the software in the presence of a customer representative and submit a copy of the code to the instructor.

Integration of research in the course

To make room for the new research component, several measures were taken. First the development project group size was expanded to six students, with one member acting as the leader and contact person for the group. We also let the same groups who handled the development projects work on the research projects. Second, a laboratory instructor, with an MS degree in computer science and SE teaching experience at a local university, was employed to assist in the coordination and evaluation of development project activities. This included performing code inspections and checking group progress. Third, we used the WebCT software [19] to design a Website (with chatting capability) for the course, which provided a medium for interaction among students or with the instructor and the laboratory instructor.

At the start of the semester, students were asked to answer a questionnaire designed to inquire into the students' backgrounds, attitudes toward research, related interests, goals and career plans. The results revealed that about half of the class had some exposure to research, while the others had little or no experience, as undergraduate research was not stressed as an essential requirement. More than 80% of those who had some research experience replied that previous research had helped them learn more about a topic, despite the difficulties they encountered due to the lack of guidance in completing their assignments. Many students demanded that more research be integrated at the undergraduate level, while some were uncertain due to the fear of becoming overwhelmed by work. Through personal contacts, many professors expressed reluctance to give undergraduate students research assignments through courses, believing that most students, driven by the grade, will end up plagiarizing without contributing to the topic. When asked about their point of view on this issue, more than two-thirds of the class said that this symptom is prevalent in situations in which students are asked to deliver research reports without having a mechanism for early feedback.

Research component design

Taking into account that most students did not have prior exposure to research and realizing that a progressive approach might culminate in an effective learning process, the course instructor chose 18 full-text journal papers tackling various angles of Open Source software (OSS) in spring 2004, Extreme Programming (XP) in fall 2005 and software engineering practices in industry in spring 2006. The course design and methodology were the same in all offerings. Hence, and to simplify our discussion in this paper, we will concentrate on discussing the work done by students in the spring 2004 class (the first time the new course design was implemented) and, at the end of the paper, we will compare overall student performance in all offerings.

A guide (below) was provided outlining the step-by-step process of conducting the research, starting from the initial stage of reading the papers and ending with writing a journal paper-like report:

- There are 18 papers related to software engineering and Open Source. These papers are zipped in the file papers.zip, which is available on WebCT
- Each group must read all 18 papers as follows:
  - Every group member should read three papers.
  - For each reviewed paper, a member summary report must be written that is one-quarter to one-third of a page long, and which highlights: (1) basic idea, approach, technique, novelty, or characteristics; (2) mentioned advantages or pluses and limitations or issues; (3) your own assessment and evaluation.
- The six group members should meet to link their summaries together into one cohesive group summary report that reads as if it was written by one person:
  - Whenever there are common features, ideas, issues among the papers, these must be compared and talked about in the same paragraph.

  - Distinct subjects must be covered in their own paragraphs but if they are related to other subjects (from other papers), then they ought to be together.
  - The report must reference the papers wherever you refer to their content. Use numbers in square brackets, e.g. [3], where 3 is the third listed paper.
  - I advise you to look at software engineering journal papers as examples. These can be found in the Engineering Library.
  - Group reports must be close to two A4 pages, single-spaced, 12-point font.
- Next, the members should discuss and brainstorm ideas for improving on the method, approach, or design that was described in one or more papers. The group should then choose one idea and write the idea report:
  - Describes the idea.
  - Describes the anticipated advantages and benefits.
  - Mentions its limitations and constraints.
- The members then should decide how to prove their idea. This can be done through a test program, code analysis, statistics, etc. Once this step is completed, the group writes the analysis report, which describes your work, experiment, data, pseudocode, etc.
- Grading will be based on the following breakdown:
  - Member summary reports: 15%.
  - Group report: 20%.
  - Idea report: 15%.
  - Analysis report: 50%.
- **Schedule:**
  - Students must follow the schedule below. Along with the last three deliverables, each group will present its progress in class within 10 minutes.
  - Member summary reports: March 16.
  - Group summary report: March 25.
  - Idea report: April 6.
  - Analysis report: May 21.

The class consisted of 36 students in the spring of 2004 (SP04), 38 students in the fall of 2005 (FA05) and 43 students in the spring of 2006 (SP06). The number of groups ranged between six (SP04 and FA05) and seven (SP06). As mentioned in the box below, each group was supposed to read the same set of 18 papers, generate a literature survey-like summary (phase 1), come up with one or more ideas that would potentially lead to improved approaches and/or concepts (phase 2) and finally, deliver a report that contains analysis and validation of the proposed approach or methodology (phase 3).

The intent behind the above arrangement was to get students started with a current and interesting SE subject and then identify a specific field of interest on their own, then they study additional papers to develop a background for their research. The overriding goal was to let students experience a sense of project ownership that will probably form a motivational factor and provide training for performance in situations similar to those encountered during the initial phases of graduate thesis work. Moreover, allowing students to choose the subject gives them an option to change topics in the early stages of the semester, which then promotes the development of additional skills for feasibility studies, risk assessments and evaluations of interests and long-term goals, among others.

Members of each group were required to present the progress of their research using PowerPoint slides as part of each phase. The importance of these presentations was not only their grade weights, but also they obliged group members to address the class, thereby honing their communication skills and proficiency. This opened the door to comments from both the class and the instructor, which was deemed vital in improving the quality of their work, as well as in keeping them on the right track.

Performance assessment

In designing the course components and devising an assessment policy that is representative of performance, the instructor drew on his 11 years of professional experience in software development. In addition, the laboratory instructor provided assistance in evaluating assignments related to the development project and in giving a second opinion on the research project reports. To ensure close to equal participation and involvement of group members in all aspects of the development project, we required students to use the Microsoft Project^{TM} software for task scheduling and tracking and submit generated Gantt charts at preset dates to the laboratory instructor for evaluating group and individual progress.

Timing of research component offering

To fulfill the Bachelor of Engineering (BE) requirements, students must spend the summer of their third year on practical training (internship). Many of those who plan to pursue graduate studies arrange to spend the summer at universities in Europe and in North America, to work on faculty research projects, mainly to increase their chances of gaining acceptance to graduate programs at those universities. The integration of research education into the third-year SE class is well timed for students, as it enables them to become more effective in their internship assignments. As for those who elect to go to the industry for training, the gained research skills can potentially improve their problem-solving skills and self-reliance, which are characteristics that employers desire.

CLASS PERFORMANCE

Research phase 1: literature summary

To ensure individual contribution, it was required that the 18 papers be divided among the members of each group as equally as possible. Each person had to read and summarize the key concepts of the three papers and submit a two-page report. The members of every group then discussed their understandings of the papers and combined their individual summaries into a coherent summary up to three pages in length. This was reportedly a major challenge, as the members had to decided the key concepts to keep and how to link them together and still point out the specific topics that the papers addressed. Furthermore, students were asked to include their remarks and provide suggestions that may represent alternative approaches.

Reports: all groups were able to report the obvious concepts that were present in the papers. More specifically (in the SP04 class), they elaborated on the positive characteristics of Open Source, namely flexibility, low cost, stability and reliability. Similarly, they reported the limitations, vulnerabilities, lack of standards, relatively poor interfaces and maintenance issues. Finally, everyone stressed the need for thorough documentation to enable developers to successfully integrate Open Source code with their own. However, when it came to the implicit and profound themes, only selected groups were able to identify and discuss these in their reports. Such topics included the role of redundancy in detecting software faults, benefiting from hacker skills in designing secure code, role of open forums in reducing the time to deployment and the views of profit-making software companies on Open Source and abandoned code.

To judge the different groups while keeping in mind that the summaries for the 18 papers were limited to two pages, we categorized the different subjects in the papers into obvious topics and implicit topics. We also added a third measure about the organization of the paper and the flow of topics presented (illustrated in Table 1). The assessment was based on the proportion of subjects that each group was able to pinpoint and on how each group structured its paper.

Presentations: the presentations were meant to get students comfortable with addressing a sizable crowd, where it is crucial that they exhibit self-confidence and the ability to capture the interest of the audience. Instead, an overall trend of inadequacy of preparation, clarity of speech and of relaying ideas was observed. The presentations, however, were very beneficial in allowing all groups, who had read the same 18 papers, to learn from each other and to identify ideas and concepts that they had missed in their own summaries.

Group members were required to present key points and evaluations of the articles that were read. They had up to half an hour (about 5 minutes per person) to assess the main topics of each paper. A close analysis of the presentations revealed two levels of differences. The first results from analyzing how each group presented the same articles and what main points were valued to be worth sharing with the class. The second was the discrepancy in style, skill and effort that arose between individuals of the same group.

Table 2 presents the performance of the six groups in the SP04 class according to three sets of criteria. The numbers revealed the consistent performance of most groups in the content categories. As for presentation skills and inner-group uniformity, there were discrepancies between groups, indicating possible differences in motivation, background, presentation and other skills. Finally, it was noted that the relatively close performance averages across all categories was mainly because most groups included a mixture of strong and mediocre performers, a fact that was reflected through the standard deviations of the last set of categories.

Finally, we combined our assessments of the individual summaries, group summaries and group presentations into one chart (Fig. 1). The objective was to investigate possible correlations between these three components and to provide a picture of the class performance for this first phase. From the figure, where the grades for the individual summaries and presentations were averaged for each group, one can observe the low or lack of correlation between the performances on all three components. It is especially surprising to observe the significant inconsistencies between the assessments of the individual summaries and group summaries. This could be attributed to the lack of coordination and organization skills within the groups. Searching for a more concrete explanation, we also included in the figure the in-group standard deviations that correspond to the presentations and individual summaries. As seen, the two curves are correlated. Since there are common elements (e.g. interpersonal skills) that affect performance in the two corresponding areas, one can relate the discrepancies in the performances for each group to a low level of homogeneity and coordination.

Table 1. Group performance in identifying and organizing topics (SP04 class)

	Group number					
Topics and paper structure	1	2	3	4	5	6
Obvious topics: characteristics, limitations and desired features	7.5	8	9	9.5	8.5	8
Implicit topics: redundancy, fault detection, hacker's experience, forums, software companies, etc.	5.5	4	6	9	6.5	7
Organization and flow	6.6	6	6.5	9	7	6.5

Table 2. Categorized group performance for presentations (SP04 class)

Criterion		Weight	Group Number 1	2	3	4	5	6	Average	Std Dev.
Content	Main ideas	7%	1	2	3	4	5	6	8.7	0.8
	Key concepts	11%	8	9	8	9	8	10	7.3	0.5
	Positives/negatives	11%	7	8	7	7	7	8	7.0	0.6
	Own evaluations	12%	6	7	7	7	7	8	7.5	0.8
Presentation	Attracting attention	5%	7	7	7	8	7	9	6.3	1.2
	Speaking skills	5%	5	8	6	7	5	7	6.0	1.1
	Feeling at ease	4%	5	6	6	6	5	8	6.5	1.5
	Illustrations	8%	6	7	5	7	5	9	7.0	1.1
	Logical transitions	8%	6	7	7	6	7	9	8.0	1.1
	Adequate length	6%	6	8	8	9	8	9	8.2	1.5
Uniformity	Content	8%	7	6	8	10	9	9	6.8	1.3
	Style	7%	5	7	7	9	7	6	7.3	1.4
Weighted average		8%	7	5	9	8	8	7	7.3	1.4

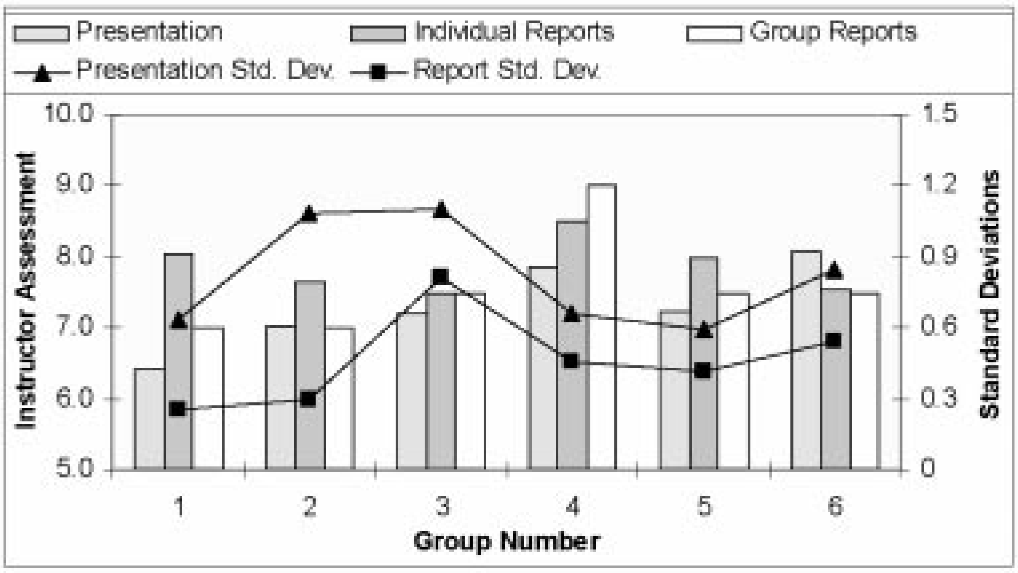

Fig. 1. Group performance of phase 1 activities (SP04 class).

Research phase 2: idea development

In this phase, students had to analyse their own idea and give a brief introduction on how they would apply it to a particular research topic. This requirement demands more expertise and beforehand preparation, as it is not sufficient to simply highlight specific concepts and present them in class. Students are now required to discuss tentative and undemonstrated theories, while attempting to convince their audience. In this phase, the groups had to both submit written idea reports and then give a presentation explaining the ideas that will form the foundation for their research. They were asked to propose improvements of already existing methodologies, new approaches, or in-depth analysis of an OSS-related topic.

Reports: the reports gave more insights and details about the proposed ideas for research. For illustration, we provide below a brief description of the proposal of each group, taken from the SP04 class:

- Group 1 proposed building a knowledge base for storing and accessing source code that is developed by students and instructors.
- Group 2 was interested in analyzing the ability of future techniques to discover and model software development processes.
- Group 3 wanted to investigate the use of Open Source in academia and the lack of documentation and non-conformance to standards that are associated with OSS.
- Group 4 proposed to research the impact of Open Source on selected software development models. The effort should focus on the state, organizational and control views of the models.
- Group 5 wanted to examine the threats that Open Source may pose on companies such as Microsoft. The work is to identify software that lends itself to Open Source and the one most suitable to be proprietary.
- Group 6 intended to examine OSS copyrights and the conditions that are placed on the use of

Open Source in developing commercial software. The group was to also report on the use of OSS in the Middle East.

The feedback that was given to groups in both offerings related to two areas: paying more attention to the analysis aspects of the study and favouring depth over breadth. For example, the above-listed subjects proposed by Groups 1, 3, 5 and 6 are not considered the most significant for software engineering, but nevertheless the instructor allowed for less related subjects, because the main objective of the research assignment was to teach students the processes of carrying out research and developing related skills, rather than solving SE problems. The goal of the development projects, on the other hand, was to have students build software systems methodologically in accordance with SE frameworks.

Presentations: this was the second required presentation and had to include the reasons behind the ideas. The objective was to force students to foresee how to implement their ideas within an actual real world context, where constraints are omnipresent issues. It was observed that students were more comfortable in presenting, for many it was the first time they had to devise a way for enhancing a methodology. As in the survey phase, all members of each group took part of the presentation. Here, however, the presentation centered around one theme and students were able to take turns in explaining different facets, scope and timeline. Because the presentations were short (about 10 minutes), students were assessed as a group, not individually, based on four main scales as indicated in Table 3. In particular, Groups 4 and 6 did a great job in defining their selected topics, identifying the aspects of their idea and outlining the plan of action they intended to follow in the research. As for the other groups, they were lacking in one or more facets.

The assessment of the groups on both the presentations and reports are shown in Fig. 2. In contrast to how the groups fared in phase 1, here it appears that a correlation exists between the performances on the presentations and on the reports. This may be indicative of the fact that student presentation skills have improved and are no more hindering the ability of the presenters to express their thoughts.

Research phase 3: analysis

For this last stage, the groups had to submit final written reports that present the analysis of

Table 3. Group performance in idea presentation (SP04 class)

Group	Subject of research	Weights				Type			
		40% Novelty info.	20% Plan info.	15% Enthusiasm	25% Clarity	System	Approach	Analysis	Weighted average
1	OSS knowledgebase	7	8	7.5	8.5	✓	✓	—	7.7
2	New OSS modeling	7	7	7.5	8.5	—	—	—	7.5
3	OSS computing center	8	7	6.5	8.5	✓	—	—	7.7
4	Affect of OSS on SE models	8	7.5	9	8	—	—	✓	8.1
5	Affect of OSS on Microsoft	6	8.5	8	9	—	—	✓	7.6
6	Open Source and copyrights	9	7	8	8	—	—	✓	8.2

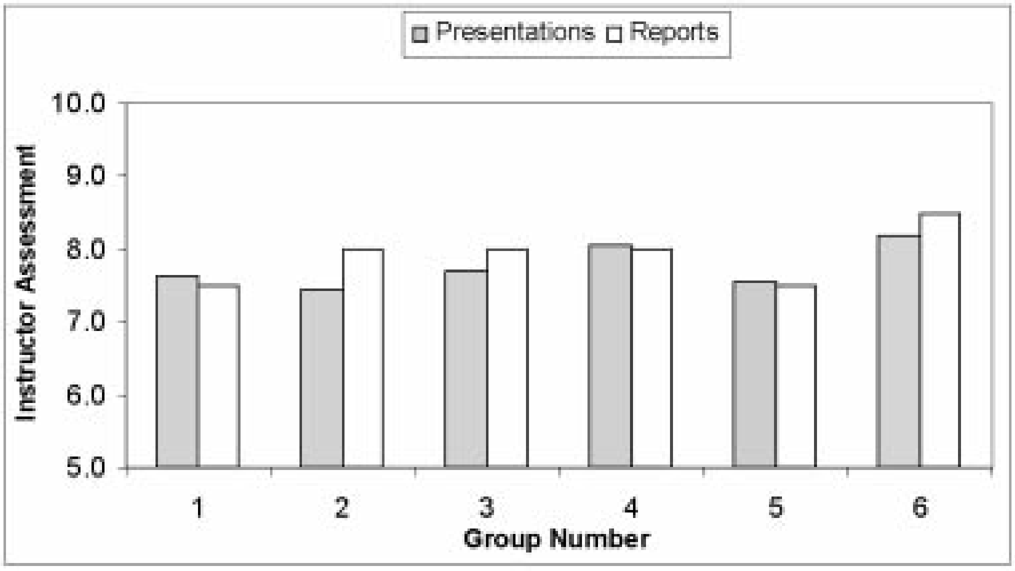

Fig. 2. Performance of groups in presenting and reporting ideas (SP04 class).

Table 4. Categorized group performance for the analysis reports (SP04 class)

	Weight					
Group	0.25 Significance	0.3 Novelty	0.2 Clarity	0.1 Structure	0.15 References	Weighted average
1	8	7	8.5	7.5	6	7.5
2	9.5	9	8.5	8.5	8.5	8.9
3	7	7.5	8	8	6	7.3
4	8.5	7	9	8	8.5	8.1
5	7	7.5	8	7	9	7.7
6	9	9.5	9	8	10	9.2

proposed approaches, results of surveys or design of systems, in addition to giving PowerPoint presentations. The majority of the grade that was designated to the research component (20% of the overall course grade) was awarded to this report. We briefly describe below the reports that were submitted by the groups in the SP04 class.

Reports: the reports of Groups 1 and 3 centered around one idea. Through surveys, they found that while working on projects, students spend much of their time building non-value-added software and that they are able to deliver more functional software if they had access to written code that implements utility functions and user interfaces.

Group 2 tackled several related topics such as the non-applicability of traditional models to OSS development and discussed the necessary changes to the Waterfall and Spiral models, dealing with the idea that OSS evolution and maturity is directly correlated to utilization and further development. Other parts of the report dealt with Open Standards, the role of UML in standardizing OSS development and distributed software development. The report may seem non-focused but the group was skillfully able to link the different subjects under the theme of OSS development.

Group 4 learned from a survey and other means the desired characteristics of models that would apply to OSS development. In addition to proposing a new model, termed the Cyclic Model, the group proposed modifications to the Waterfall, Prototype, Spiral, Transformational and V models, mostly by incorporating feedback paths from the testing and maintenance stages to the design stages. The report concludes by discussing the applicability of the models to a senior-level software project.

The report of Group 5 presents findings that point out a general trend in which developers increasingly integrate OSS in building software solutions. The report also cites cases in which management opts for commercial solutions that are supported by big companies, such as Microsoft, due to factors relating to the criticality of customer service and familiarity of employees with commercial software, e.g. Windows.

The report of Group 6 addressed the issue of OSS copyright. It contrasted commercial software licensing to freeware licensing and discussed the differentiating characteristics of some well known OSS public licenses. Reported statistics depicted the status of the legal systems, as they relate to intellectual property protection, in five Arab countries. It is worth noting that the group obtained this information through interviews with prominent lawyers and professors of law and through digging into news archives.

Having explained the topics tackled by the different reports, we now present the evaluations that were based on several criteria, as shown in Table 4.

The average quality of the reports was reasonably good and it was obvious that almost every group put in considerable effort, including conducting surveys, holding interviews and carrying out investigations. Groups 6 and 2 were the best performers, in that their work was original, informative and well presented. The majority of groups resorted to surveys and/or personal interviews. Some used these to learn about the level of utilization of certain techniques (Group 4), others wanted to get a feel for the need for particular capabilities (Groups 1 and 3), while the rest were seeking specific information relating to the status of software sharing and copyrights (Group 6). All groups, but one, came out with recommendations, new approaches or designs. Finally, it is worth elaborating on the References criterion in Table 4, which was used to evaluate the use of additional references (other than the 18 papers referenced above) by the groups. Groups 1 and 3, each cited two conference papers and two Web documents. Groups 2 and 4 were also comparable in this respect as each group cited three additional journal papers, two conference papers and one Web document (Group 2 only). Group 5 cited eight Web documents and five technical reports. Finally, Group 6 topped the scale by citing five additional journal papers, four technical reports, two news articles and two Web documents. In all, four of the six groups did a decent job in finding related work and in using it to support their research.

Presentations: the members of each group had to decide how to explain their research and findings effectively and devise a strategy for making the transition between members seamless and smooth. Every group had 20 minutes of presentation time in addition to 10 minutes for questions and answers. Having presented in the class twice before, it was obvious that most students felt more comfortable when carrying out their parts of the presentation. For evaluation, five general criteria were considered when assigning points to students. These concerned their ability to explain the research background, to specify their contributions, to portray the significance of results or analysis, to discuss the implementation issues that were encountered and to answer questions effectively. As each student was handling one part of the presentation, not all criteria were applicable. The evaluations are presented in Fig. 3. This figure shows a clear correlation between performances on the reports and presentations. The within-group standard deviations of presentation performance are moderately low and do not fluctuate significantly across groups. One may conclude that, after few presentations (three to four), a class such as this (third year Computer Engineering) may reach a steady level of presentation skill variability across all students. Any remaining variability could be tied to external factors such as student background and motivation.

IMPACT AND ASSESSMENT OF LEARNING

Many researchers reported on the benefits of the undergraduate research programs, such as students obtaining employment [7], publishing papers in conference proceedings and journals [20, 6], or pursuing graduate studies [20, 2]. To study the impact of integrating a research component into the undergraduate software engineering course, we employed several measures. First, we compared the assessment of student work in all three course offerings that were mentioned above, to ensure that the discussed results are not accidental. Second, we analyzed student assessments and examined the feedback obtained from students at the end of the semester. Third, we studied the incurred cost by examining the time spent on the additional course component from the student and instructor point of views. Finally, we looked at the annual exit survey results conducted by the university's Career Center, which compiles statistics about graduating students. Of particular interest to our study was the proportion of ECE undergraduate students who opted to pursue graduate studies versus those who chose to start their professional career after getting their BE degrees in the years 2004 and 2005, in order to see if the SE course could have influenced this proportion.

Student learning

When considering the reports, one notices that they differed in terms of purpose, content and needed skills. Furthermore, the first and second reports were, in the most part, very short relative to the third in all three classes. This made it impossible to infer student (group) learning from the three reports.

The presentations, however, share many common features, thus allowing us to derive learning trends. For this purpose, we plotted the three phase assessments for each group of the three classes (in the top part of Fig. 4, averages across students were computed for phases 1 and 3), in addition to the trends across all groups in the bottom part. As shown, all groups show steady improvements, albeit to varying degrees. In general, groups that started out the lowest improved the most when going from phase 1 to phase 2. Also, it is interesting to observe that all but one group in the FA05 class were able to attain almost the same high level of presentation skills by the end of the semester. Part (d) of the figure plots the linear trend curves that were fitted to the

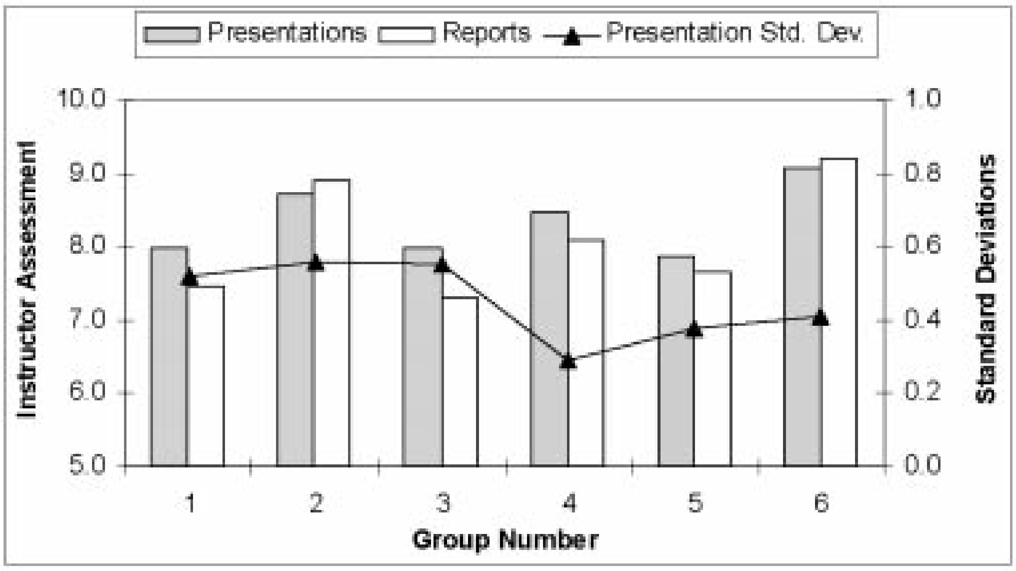

Fig. 3. Group performance in the analysis phase of the research (SP04 class).

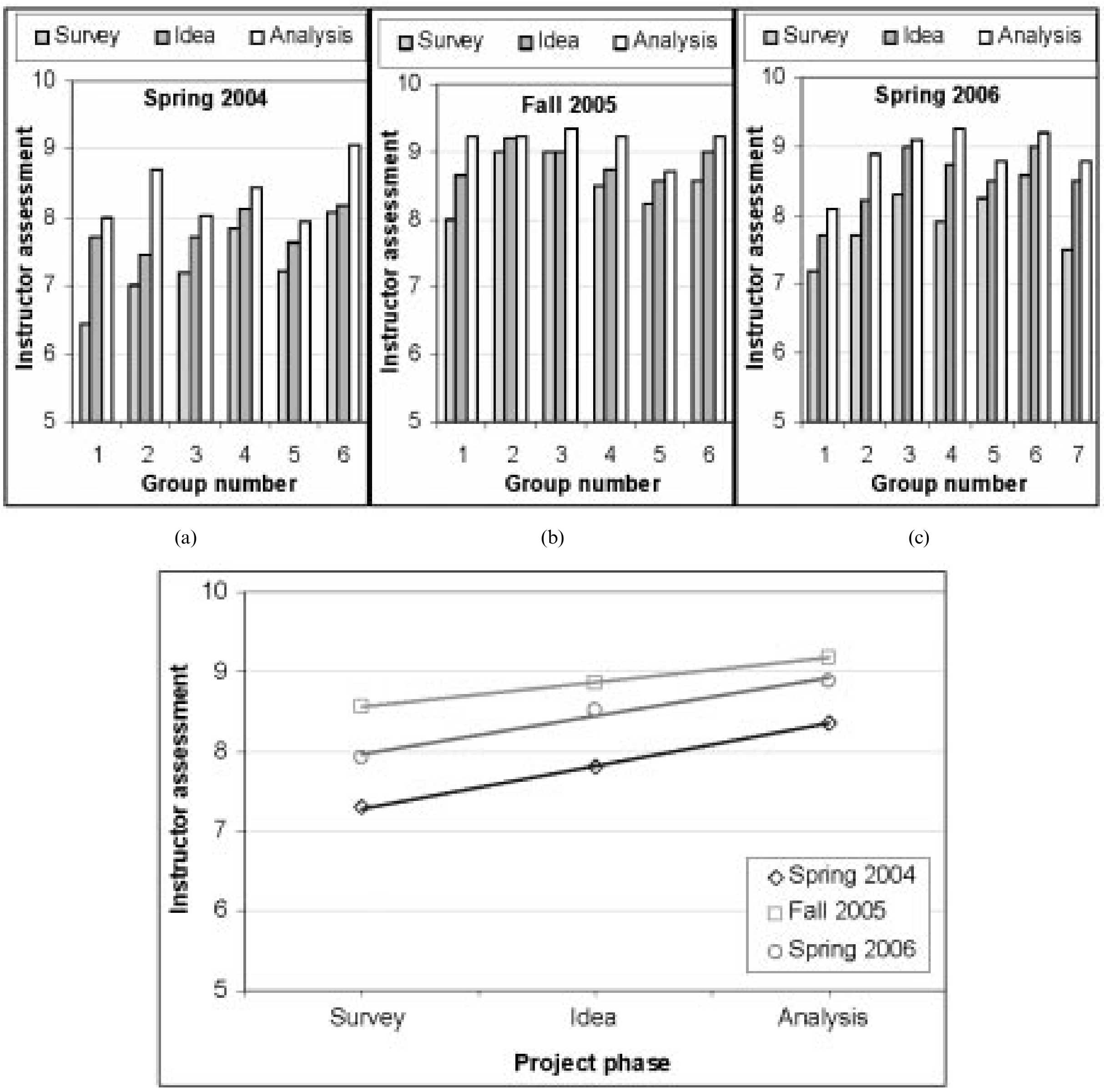

Fig. 4. Individual group performance on presentations (a–c) and trends over all groups (d).

averages for each class taken across groups. The slopes of the curves ranged between 0.31 (FA05) and 0.53 (SP06). Students improved by an average of four points through each phase transition (a total of 8 points throughout the course).

Student assessments

At the conclusion of the course, students were asked to evaluate what they had learned through the research part of the course. This was not a multiple choice questionnaire, but rather an open forum through which students expressed their appraisals and thoughts in their own words. Judging from the feedback (from a total of 108 students in all three classes: an 84% response rate), we were able to conclude that the added research component in the SE course helped students to:

- Acquire knowledge and skills in an academic field that transcends traditional classroom study.
- Clarify academic and career interests and goals.
- Gather new knowledge about current topics in software engineering research.
- Enhance skills in communication, independent thinking and problem solving.
- Promote experience in performing research, conducting presentations, writing technical papers and doing group work.
- Increase professional and academic credentials to support applications for scholarships, awards, career employment and entry into graduate schools.

The above points provide an indication of the amount and type of benefits that students received from this class experience. Not only does it show that skills were enhanced and experience was gained, but there is a sense of benefiting toward reaching career goals.

Cost for adding the research component into the course

When asked, close to 40 randomly selected students from the three class offerings estimated the weekly time it took them to work on the course and on the research component. According to the numbers, the average weekly time for the course was 8.3 hours with a standard deviation of 2.1, out of which the research component consumed an average of 3.4 hours with a standard deviation of 1.2. Students pointed out that the load of the course is high, when compared to that of a typical third year technical course, which was estimated at close to 6 hours per week. We note that the average load of a third year computer engineering student is five lecture courses plus a laboratory course, totalling 16 credit hours per semester. It is also worth noting that most students end up mixing technical courses with one or two humanity-type courses, which demand much less time, thus enabling them to commit more time to the technical courses. Another factor that helped students with all the components of the course was the large group size, consisting of six students in most groups, which reduced the average load on each student while working on both the research project and the software development project (discussed below).

As to the additional load on the instructor, the research component required regular follow-up, in addition to reading the reports and giving timely feedback. Given the help that was received from the experienced laboratory instructor in coordinating and assessing the development project activities and providing a second opinion on the research reports, the time that was required to run the course was still manageable.

Influencing career choices after graduation

An interesting finding would be to understand the potential impact of the introduced research component of the SE class on the decision of students for choosing the type of career path after graduation. More specifically, we are interested in learning about the proportion of students who elected to pursue graduate studies to those who opted to start a professional career, before revising the SE course and afterwards. Unfortunately, at the time of writing this paper we were only able to acquire two data sets from the university's Career Center for surveys conducted in September of 2004 (concerns graduating students who took the older version of SE) and September of 2005 (covers students who enrolled in the spring 2004 class). The survey results included the student names, so we were able to identify those who took SE classes (at AUB students should take SE or Operating Systems in order to graduate). In addition to having only two data points, we recognize that many other factors (i.e. besides being exposed to research during undergraduate studies) affect students' decisions about career paths. Nevertheless, we mention the survey data, given that there is a possibility that serious exposure to research has indeed affected student decisions. The data is presented in Table 5, which shows that the number of students who chose to go into graduate studies increased by more than 10% going from 2004 to 2005.

Table 5. Plans after graduation for ECE students

	Year 2004	Year 2005
Graduate studies	42.67%	47.33%
Employment	47.63%	45.2%
Graduate studies and employment	9.7%	7.47%

Assessing individual contributions to group work

This section discerns the research project tasks that were performed by individual members of the group and by the group as a whole. Ensuring that members of the group put in equal effort (in the approximate sense) was a key objective, so as to distribute the load fairly among members and to correctly assess the efforts of each member. Table 6 summarizes the individual and group responsibilities toward completing the research project. Making students aware of their responsibilities at the onset of the semester has helped in avoiding major issues relating to lack of participation in-group activities. It helps also to mention that in regards to the development project, the required

Table 6. Member and group tasks

	Phase 1: literature summary			Phase 2: idea development	Phase 3: analysis
	Reading papers	Summary report	Presentation	Idea report/ presentation	Analysis report/ presentation
Member	Read 3 papers	2 page summary report	Present his/her own part	Present a part of the idea	Present a part of the research and findings
Group	Discuss 18 papers	3 page summary report	Manage coherent presentation	Submit one common idea report/ presentation	Submit one common analysis report/ presentation

Table 7. Elements of the development project

Project element	Book chapters	Weight (%)
Project description report	—	5
Activity list, effort estimates, preliminary schedule and personnel assignments	3, 4	10
Functional diagrams and pseudo code	5, OM	7
Design complexity report	5	8
UML diagrams	6	10
Project meetings, code inspections and Gantt charts	7, OM	20
Fault log and fault report	8, 9	10
Demonstration of developed system	—	30

Gantt charts that had to be submitted by students after the completion of each major milestone have greatly helped in keeping track of individual efforts and contributions. Finally, it was natural for the instructor to consider his evaluation of individual contribution to the group effort when awarding grades to work on the projects.

SOFTWARE DEVELOPMENT PROJECT COMPONENT

Although it is not the focus of this paper, we discuss briefly the software development project, which is the most important element in the SE course since it allows students to apply the learned SE concepts while developing a practical database-driven software system. The elements of this project are shown in Table 7 along with the corresponding book chapters and the weights for calculating the total grade for the project, which constituted 40% of the overall course grade. The shown symbol 'OM' denotes outside material that was used as a supplement to the book. An example is the part of the lecture that was spent on explaining the many forms of pseudo code, or the two hands-on sessions on using the Microsoft Project software (for task scheduling and progress tracking) and using the Visio software (for generating UML diagrams).

As we have done earlier, we provide a list and brief description of the projects that were done by students in the spring 2004 class. The list is depicted in Table 8.

The first project is related to the group's research, where the lab instructor acted as the customer. The third project was an implementation of a Web page change detection system that was described in a journal paper. The customer for the group was a graduate student who wanted to compare the performance of his approach to the one in the paper. The remaining projects were done for organizations that actually needed the systems.

The members of Group 1 struggled initially to agree on a topic for their development project but finally decided to choose the implementation of their research idea as the subject. These students indicated that they worked on the course requirements for an average of six and a half hours per week, which was the lowest among all groups. Although combining the development and research project is expected to save time, students had to wait for the completion of the investigation stage, which includes surveying prior work, thinking about a topic and deciding what needs to be developed. Actually, the developed system of Group 1 was missing functionalities that were specified in the design requirements, implying that the members ran out of time.

The third development project was the most demanding. Students in the group estimated that they spent more than 10 hours per week toward the course, which was the result of agreeing to a full implementation of a system that was not completely defined. They were able to deliver a complete system but this may have been at the expense of their performance in other courses, as one student stated. This however was totally driven by the students' interest in the subject they worked on.

As for student performance, each group completed the tasks shown in Table 7. The demonstration, which was done at the end of the semester, was conducted in the presence of the customer and focused on the functionality, usability and robustness of the developed system. The total project grade was a weighted average of the scores on the individual elements in accordance with the weights shown in Table 7 (shown in Fig. 5). Group 3 of the SP04 class received nearly a perfect score due to the quality and the size of the completed work. Most importantly, the group was able to deal with incomplete specifications and develop an entire Web page detection that was only missing a crawler. Next, from the same class, it was group 2 that developed a practical system for a local pharmacy, which was highly praised by the customer. Group 5 received the lowest score, as the developed system was functional, but scored poorly on the usability scale. The remaining groups did fairly well as they were able to develop systems that met the requirements and got positive remarks from their customers.

Table 8. List of student development projects

Group	Short project description
1	Knowledge base system for educational use
2	Inventory and payroll system for a pharmacy
3	Web page change detection system
4	Delivery tracking system for a food distribution company
5	Online hiring system for a contracting company
6	Database management and synchronization system

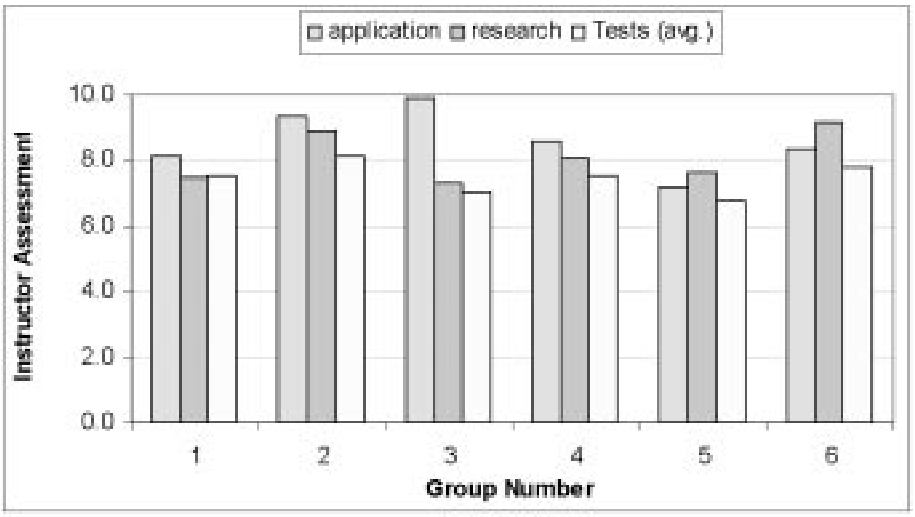

Fig. 5. Group performance on the development project.

CONCLUSION AND LESSONS LEARNED

Through the addition of the research component to the Software Engineering course, students were able to enhance their learning by performing real research and tackling professional issues such as conducting presentations and participating in group work. Despite the necessity to commit more time, the students' successful performances in traditional components of the course showed that integrating research into the program did not adversely impact their overall output. As a remedial, one area that may be improved in future offerings is to make provisions related to the research subject, logistics and perhaps the development project in order to reduce the time required from students to complete the requirements, without compromising the objectives. For instance, before the start of the semester, delegating the task of finding development projects to the lab instructor or the Career Center could save students time by enabling them to select and control the scope and size of the projects beforehand.

To investigate possible connections between the performance of students on the tests, development project and research projects, we computed the correlation coefficients between the performance of the class on the research project and its performance on tests on one hand and its performance on the development project on the other hand. These coefficients, which are plotted in Fig. 6, suggest a relationship between performance on tests and performance in research and a stronger relationship between performance on the development project and that on the research project. This shows that students in general were able to dedicate the needed effort to all the components of the course and obviously find the time to do it. The

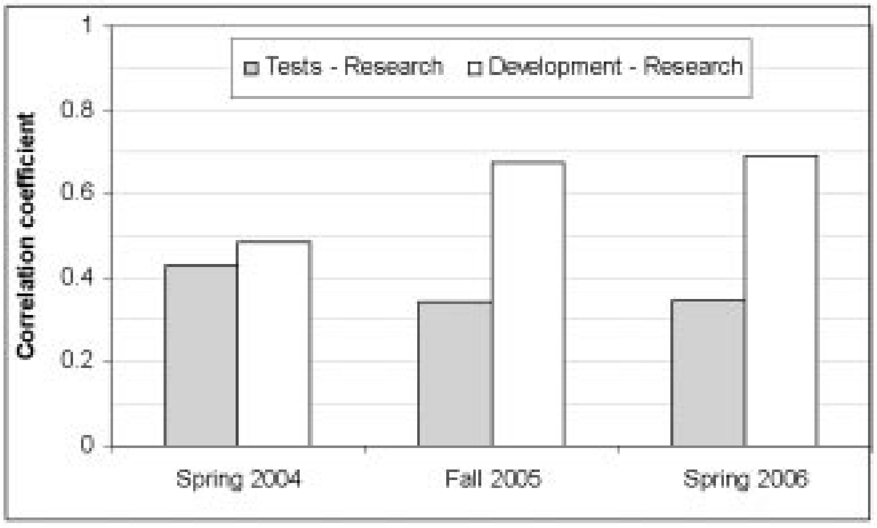

Fig. 6. Coefficient values for correlation between performance on tests and research and between performance on software development and research.

lack of a perfect correlation implies that the three elements (tests, development and research) require different sets of skills and serve to accomplish complementary objectives.

Finally, it should be mentioned that we examined the curriculum proposed by Bagert *et al.* [21], which incorporates suggested material for SE classes. It breaks the SE body of knowledge into four areas: core, foundations, recurring and supporting. A research component could be added to the recurring area, which includes components that may be encountered throughout the core area phases. The goal is to equip students with skills that would enable them to adapt learned concepts to the continuously evolving nature of practical software development (e.g. Open Source, Extreme Programming, etc.). The research component of the SE course is expected to promote the investigative and problem analysis skills of students and raise their awareness of the large body of research literature that treats new SE methodologies and techniques.

REFERENCES

1. A. Lopez and K. Messa, An undergraduate research program in multi-paradigm software design, *Proc. of the SIGCSE Technical Computer Science Education Symp.*, **26**(1), 1994, pp. 271–275.
2. S. Humphreys, Summer Undergraduate Program in Engineering Research at Berkeley, *Proc. of the Teaching and Learning in an Era of Change Conf.,* **3**, 1997, pp. 1137–1139.
3. E. Bogucz and E. Spina, Comprehensive undergraduate research program as a sample graduate school experience, *Proc. of the ASEE Conf.,* **1**(1), 1995, pp. 1–4.
4. D. Sabatini, Teaching and research synergism: the undergraduate research experience**,** *Journal of Professional Issues in Engineering Education and Practice***, 123**(3), 1997, pp. 98–102.
5. P. Johann and F. Turbak, Lumberjack summer camp: a cross-institutional undergraduate research experience in computer science, *Computer Science Education*, **11**(4), 2001, pp. 279–304.
6. H. Davoodi, F. Just, A. Saffar and M. Noori, Collaborative undergraduate research, *Proc. of the Frontiers in Education Conf.,* **3**, 1999, pp. 19–20.
7. M. Kurland and H. Rawicz, Involving students in undergraduate research and development: two perspectives, *Proc. of the Engineering Education for the 21st Century Conf.,* **2**, 1995, pp. 1–6.
8. *Reviews in Undergraduate Research*, 2006. [online] available: http://www.ruf.rice.edu/~rur/
9. *Caltech Undergraduate Research Journal*, 2006. [online] available: http://www.curj.caltech.edu/front/indexUC.php
10. *Council on Undergraduate Research*, 2004. [online] available: http://www.cur.org/
11. *National Conferences on Undergraduate Research*, 2006. [online] available: http://www.ncur.org/
12. Research Experiences for Undergraduates, *National Science Foundation (NSF),* 2005. [online] available: http://www.nsf.gov/pubs/2005/nsf05592/nsf05592.htm
13. N. Passos, Software engineering requirements, analysis in the classroom, *Journal of Computing in Small Colleges*, **12**(4), 1997, pp. 48–57.
14. N. Passos and S. Carpenter, True undergraduate research: foundation for graduate studies and critical thinking, *Proc. of the Frontiers in Education Conf.,* **3**, 1999, pp. 7–12.
15. S. Jovalekic, Project-oriented approach to software engineering education in a multidisciplinary environment: objectives, realization, evaluation, *Proc. of the Frontiers in Education Conf.,* **2**, 1996, pp. 501–505.
16. University of Illinois at Urbana-Champaign, ECE Illinois, http://courses.ece.uiuc.edu/ece497/index.asp#497
17. R. Greendyke, Graduate level research from undergraduate students: the lessons learned by student and professor alike, *Proc. of the Frontiers in Education Conf.,* **3**, 2002, pp. S4C/1–6.
18. S. Pfleeger, *Software Engineering Theory and Practice*, 2nd ed. Prentice Hall, Upper Saddle River, NJ (2001).
19. WebCT Inc. *WebCT Campus Edition datasheet*, 2004. [online] available: http://www.webct.com/
20. M. Shah and K. Bowyer, Mentoring undergraduates in computer vision research, *IEEE Transactions on Education*, **44**(3), 2001, pp. 252–257.
21. D. Bagert, T. Hilburn, G. Hislop, M. Lutz, M. McCracken and S. Mengel, *Guidelines for Software Engineering Education* Version 1.0, Technical report CMU/SEI-99-TR-032, Carnegie Mellon University (1999).

Hassan Artail, worked as a system development supervisor at the Scientific Laboratories of DaimlerChrysler, Michigan, before joining AUB in 2001. At DaimlerChrysler, he worked for 11 years in the field of software and system development for vehicle testing applications, covering the areas of instrument control, computer networking, distributed computing, data acquisition and data processing. He obtained BS and MS degrees in Electrical Engineering from the University of Detroit in 1985 and 1986 respectively and a Ph.D. from Wayne State University in 1999. He is currently an Associate Professor and is carrying out research in the areas of Internet and mobile computing, distributed systems, ad hoc networks and data management, in addition to computer and network security.