IT Systems Development: An IS Curricula Course that Combines Best Practices of Project Management and Software Engineering

Abdallah Tubaishat
College of Information Technology, Zayed University, Abu Dhabi, United Arab Emirates

Abdallah.Tubaishat@zu.ac.ae

Abstract

Most computing programs still devote little time to software life cycle development, software processes, quality issues, team skills, and other areas of software engineering essential to effective commercial software development. A teaching project was developed and implemented by accommodating knowledge and practices that are applicable to most projects in the area of project management and in the area of software development to Information Systems (IS) students. This approach is relevant to IS model curricula and is in accordance with IS2002.10 project management and practice course guidelines. The rationale behind this approach is to overcome the relative lack of experience of IS students in many aspects of project management and software development by introducing them how to plan, organize, and control software development projects, and to help students strengthen good software engineering practices prior to entering IT industry and become more efficient. We present results of a case study based on a survey conducted in an IT Systems Development course. Survey results show that including topics on project management and Software Engineering best practices lab into an IT Systems Development course helped students (a) deal with non-technical issues of software projects, (b) develop their ability to communicate clearly with team members, and (c) overcome their lack of experience in many aspects of project management and software development.

Keywords: IS2002 Model Curricula, Software Engineering Body of Knowledge, Software Engineering Best Practices, Rational Suite Enterprise.

Software Engineering in IS Curricula

Software engineering course is taught to higher education students majoring in Computer Science (CS), Computer Engineering (CE), and Software Engineering (SE). Software engineering course is also taught in other disciplines, either as a mandatory or as an elective course, such as Information Systems (IS). IS is a broader field than CS and includes parts of CS. IS field could be described as an interdisplinary field that studies the design and use of information systems in a social context. As noted in IS2002 model curricula (Gorgone et al., 2002) , IS as a field of academic study exists under a variety of at least thirteen (13) different curricula, including Information Systems, Management Information Systems, Computer Information Systems, Information Management, Business Information Systems, Informatics, Information Resources Management, Information Technology, Information Technology Systems, Information Technology Resources Management, Accounting Information Systems, Information Science, and Information and Quantitative Science.

The author's early experience was that teaching IS students a software engineering course in the same way as CS students was not successful. This is mainly because IS students have significantly less background in programming than CS students. This experience encouraged him to accommodate topics on project management and SE best practices lab using Rational Suite Enterprise (Rational Suite Enterprise, 2008). This new approach was relevant to IS curricula and with accordance with IS2002.10 project management and practice course guidelines.

Hilburn, Bagert, Mengel, & Oexmann (2008) proposed that several computing associations including the Association of Computing Machinery (ACM), the IEEE Computer Society (IEEE-CS), and the Computer Sciences Accreditation Board (CSAB) have provided encouragement, support, and guidance in developing quality curricula that are viable and dynamic. However, most computing programs still devote little time to software life cycle development, software processes, quality issues, team skills, and other areas of software engineering essentials to effective commercial software development. Hence, new graduates know little about what are "best practices" in software engineering profession (e.g., practices related to use of software processes, team building, front-end development). Therefore, it is the role of faculty members teaching such courses to redesign and implement curricula that focus on "practice" of software engineering, and other related issues.

This paper is organized as follows: Section 2 presents arguments for the alternative approach. Section 3 presents IS2002.10 course specifications. Section 4 presents IS software engineering body of knowledge. Section 5 presents the project component, Section 6 presents a mapping from IS2002.10 course specification onto the IS software engineering course. Section 7 presents evaluation of the proposed approach. Finally, conclusions are presented in Section 8.

Why IT Systems Development Course?

We have taught the IT Systems Development course to IS students for seven years, and we believe we hit upon an approach that works. Instead of trying to instruct students in theory of various techniques, we teach them what we believe of as software development.

From the management side IS students are expected to deal with non-technical challenges arising from project situations, including understand project domain and requirements, how to be a team player, how to schedule work between team members, and how to cope with time pressures and hard deadlines. As indicated by (Weaver, 2004), students often have limited experience in projects management. They do not appreciate the need for planning and take more time than anticipated to complete tasks. We have developed the creation of a set of "guidelines" for accommodating topics on project management to help students deal with non-technical issues of software development.

From the technical side we have created a set of guidelines for accommodating SE best practices lab using Rational Suite Enterprise (Rational, 2008) to help students learn "best practices" in software engineering profession.

Our approach attempts to help students work on real world problems with the latest software tools. The rationale behind this approach is to overcome the relative lack of experience of IS students in many aspects of project management and software development, and to help students strengthen good SE practices prior to entering IT industry and become more efficient.

IS2002.10 Course Specification

The architecture of IS model curricula (IS Model Curriculum, 2002) consists of five curriculum presentation areas: IS fundamentals; IS theory and practice; information technology; IS development; and IS deployment and management processes. The five presentation areas consist of ten courses and one prerequisites course. These courses are defined at a high level with course descriptions. The IS2002.10 project management and practice course description (IS Model Curriculum, 2002) is shown in Table 1.

Table 1. IS2002.10 Project Management and Practice Course Description

Catalog:

Advanced IS majors operating as a high-performance team will engage in and complete the design and implementation of a significant information system. Project management, management of IS function, and systems integration will be components of the project experience.

Scope:

This course covers factors necessary for successful management of information systems development or enhancement projects. Both technical and behavioral aspects of project management are applied within the context of an information systems development project.

Topics:

Managing the life cycle: requirements determination, design, implementation, system and database integration issues; network management; project tracking, metrics, and system performance evaluation; managing expectation of managers, clients, team members, and others; determining skill requirements and staffing; cost-effective analysis; reporting and presentation techniques; management of behavioral and technical aspects of the project; change management. Software tools for project tracking and monitoring. Team collaboration techniques and tools.

Discussion:

This is the capstone course for IS majors who have completed the systems analysis and design sequences. It focuses on engaging in and completing a major system development project.

Within the project context management of IS, systems integration is an explicit requirement of students to address.

The project is a team effort and allows a final opportunity to practice personal and interdependence skills to ensure team member empowerment and success. Project management tools will be employed by the team to ensure tracking of the project goals and accomplishments of the client.

Automated development tools may or may not be used depending on available resources. However, standards will be developed for all project deliverables. Software quality assurance methodologies will be employed to ensure successful outcome of the project.

On going presentation of project planning, analysis, design, conversion plan, and other documentation will be done by the team. Each team member should play a significant role in some aspect of presentation.

The Proposed IT Systems Development Course Body of Knowledge

The IT Systems Development course body of knowledge taught by the author is outlined in Table 2. General guidelines for developing the syllabus for this course are taken from the well respected SE text by Summerville (2006). The IT Systems Development lab body of knowledge is outlined in Table 3. General guidelines for developing the syllabus for this lab are related to Rational Suite Enterprise (2008).

As stated in IS2002 model curricula, “The learning goals, objectives, and body of knowledge elements with competency levels provide faculty members with details of the rationale supporting the course. They provide a basis for curriculum tailoring and curriculum experimentation”. With this in mind, the IT Systems Development course taught by the author has a depth of knowledge level of achievement in the course learning outcomes according to Zayed University (ZU) Academic Program Model (Academic Program Model, 2008). It is to be noted that ZU has a strong focus on the students’ learning outcomes to improve both curriculum and learning practices. The Academic Program Model developed by faculty, emphasizes on a commitment to learner-based education and to a shift in the paradigm from teaching to learning. This model focuses on what students can actually do after they graduate.

The Zayed University Learning outcomes (ZULOs) defined for this course with their associated level of achievements (beginning, developing, and accomplished) are:

1. Critical Thinking and Reasoning.

This ZULO is addressed through the activities of analyzing different alternatives for solving problems.

Achievement level: *Developing*

2. Information Technology.

Students increase their skills in using a commonly used software packages to produce accurate and complete system specifications.

Achievement level: *Developing*

The Major Learning outcomes (MALOs) defined for this course with their associated level of achievements are:

1. Problem Identification and Analysis.

The course provides tools and techniques for analyzing problems by studying and analyzing various user requirements and then representing these requirements as a complete set of software specifications.

Achievement level: *Developing*

2. Technical Communication.

Producing well-documented requirements specifications according to current industry practice is a major emphasis of the course.

Achievement level: *Developing*

This course has been taught in this fashion five times, compared to three offering using the earlier approach. The methodology taken in this course works favorably to produce the intended outcome of the course (See the section on evaluation of the proposed approach.).

Students were engaged in and completed small-scale system development projects as one component of the IT Systems Development course. Automated development tool has been used successfully in this course.

Table 2. Syllabus for IT Systems Development Course

Course Objectives:

This course provides a fundamental understanding of:
- Project management techniques
- Various software lifecycle processes
- Use the tools provided by *Rational suite Enterprise* to apply some of the best practices of SE to automate the SE process

Topics:

2.1 Introduction (Chapter 1)
- What is software?
- What is software engineering?
- FAQs about software engineering
- Professional & ethical responsibility

2.2 Computer-based system engineering (Chapter2)
- Emergent system properties
- Systems & their environment
- System modeling
- The system engineering process
- System procurement

2.3 Software processes (Chapter 3)
- Software process models
- Process iteration
- Software specification
- Software design & implementation
- Software validation
- Software evolution
- Automated process support

2.4 Project management (Chapter 4)
- Management activities
- Project planning
- Project scheduling
- Risk management

2.5 Managing People (Chapter 22)
- How to be a team player
- Group working
- Choosing and keeping people

2.6 Software requirements (Chapter 5)
- Functional & non-functional requirements
- User requirements
- System requirements
- The software requirements document

2.7 Requirements engineering process (Chapter 6)
- Feasibility studies
- Requirements elicitation & analysis
- Requirements validation
- Requirements management

2.8 System models (Chapter 7)
- Context models
- Behavioral models
- Data models
- Object models
- CASE workbenches

2.9 Software prototyping (Chapter 8)
- Prototyping in the software process
- Rapid prototyping techniques
- User interface prototyping

2.10 Architectural design (Chapter 10)
- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

2.11 Object-oriented design (Chapter 12)
- Objects & object classes
- An object-oriented design process
- Design evolution

The IT Systems Development lab body of knowledge is outlined in Table 3.

Table 3. Syllabus for IT Systems Development Lab

The IT Systems Development Best Practices Lab introduce students to:

- Best practices in project management by presenting a framework for managing software development projects, providing practical guidelines for planning, executing and monitoring projects, and providing framework for managing risks.

- Recommended best practices of the software development process, requirement management, and use-case modeling.

Topics:

3. 1 Project Management
Introduction to project management framework
presented by the Rational Unified Process (RUP)
3.1.1 Software development plan
3.1.2 Risk list
3.1.3 Risk management plan

3.2 Software Development Best Practices
3.2.1 Introduction to recommended software development best practices
3.2.2 Overview of the RUP tools to apply best practices

3.3 Requirement Management:
3.3.1 Identify requirement and requirement management
3.3.2 Identify use-cases and actors
3.3.3 How to identify actors
3.3.4 How to find use-cases

3.4 Model Visually:
3.4.1 Introducing UML language
3.4.2 Model the diagram using Rational Rose

3.5 Introducing RequisitePro:
3.5.1 How to capture, specify and document requirements using Rational RequisitePro.
3.5.2 How to create the different project artifacts such as: vision document, glossary document, etc.
3.5.3 How to manage changing requirements using requirements traceability feature of RequisitePro.
3.6 RequisitePro Integration with Rational Rose and RUP.

The Project Component

One main component of this course is a group project. The model suggested by the author, is to ask students to develop a small-scale system by following the software life cycle in designing a solution to an information-based problem and to work in teams to implement that solution. To start a group project, students have to determine the methods used in developing projects, choose the topic, and figure out how to organize groups, and then design and implement their projects. The author spent considerable amount of time discussing methods of designing and implementing projects. Students are encouraged to put more time into considering user interface design. As an example in a course submission system, we focused on how to design an easy to use interface that help instructor manage the electronic submission of students assignments, and how to manage the grading of assignments.

Communication between group members is a required component for each project. The author followed the work done by each group by reading group meetings minutes. One meeting is set for each group once a week for a total of ten meetings in a ten-week semester. Guidelines for project work used in the IT Systems Development course are outlined in Table 4.

Table 4. Project Work

The projects suggested for the IT Systems Development Course are designed to familiarize students with the major phases of the software life cycle. It is also designed to help students to learn to work as part of a software development team. Each team will develop a system proceeding through the software life cycle. Group teams will consist of 2 to 4 members depending on the class size.

A detailed description of the *deliverables* required for each phase will be outlined below. Each team will submit the required items as a whole. These represent your cooperative efforts; the entire team receives the same grade for each item. Each deliverable must include work breakdown data. That is, for each team member a percentage figure should appear that represents the effort put fourth for this part of the project. This data must be inspected by all the team members and the team member must indicate her agreement with the assessment by writing her initials next to the appropriate entry. The amount of effort as indicated in the percentage figure will determine how much credit each individual team member receives for a particular deliverable. For example, if a team has four members and each contributes 25% to a deliverable, each will get full credit for this deliverable; the actual points depend on the grade. However, if one team member only contributes 5% to the effort, the number of points recorded for that person will be reduced in proportion to the reduction in effort.

It is important to get started on the project item as soon as it is assigned and to decide on the team organization. Remember the objective of *teamwork* is to achieve a goal with a fair division of labor. Team organization promotes the division and delegation of the various tasks to achieve the goal. Ideally, it would seem that a democratic organization would be best. This approach is the basis of the so-called “Egoless” Programming team organization. Studies have shown some kind of hierarchical organization works best especially for student groups since it centralizes decision-making and provide structure.

Using the hierarchical approach requires a *team leader* to be chosen who is responsible for defining and setting goals, organizing the group, delegating responsibilities, monitoring progress as well as performing technical tasks. Under the team leader are the technical leaders who are in charge of separate aspects of the system for each phase. Each person on a team in CIS 485 will have an opportunity to be the leader responsible for at least one deliverable.

Each of you can decide on your own team membership, which will be in effect throughout the semester. When choosing a team keep in mind that you will need to work well with each other and meet a least once a week. Once the team leader for a particular phase has been chosen, you may organize yourselves into roles you see fit.

You will probably meet with your team members at least once a week. Agreeing on a fixed meeting time when all can attend is a good policy. It is important to organize your meetings for maximum efficiency. Your team administrator should notify each member of the meeting and make sure that an *agenda* has been set up by the team leaders. Each meeting should consist of 3 parts. These are:

Preparation: decide the function and purpose of the meeting, notify members, explain agenda and make sure each team member has done their homework.

The meeting itself: focus on goals, use brainstorming to kick around different approaches, write them down, evaluate them and pick a good solution from amongst them, record events.

Follow-up: All members should understanding outcomes, what assignments they have, when they are due.

The project organization in CIT 475 is very different than you have had in your other courses at Zayed University; you will learn many new skills. Hopefully you will all enjoy and profit from this experience. To get the maximum benefits from this course you will have to be organized, cooperative and hard working.

Small Scale Projects

Small Scale Projects are those projects that a group might take through from initial specification to implementation. The following deliverables are expected from each project group:

- A project Plan and Schedule
- Vision Document
- System Use Cases

- Subsystems Scenarios - System User Interface

Mapping IS2002.10 Course Specification to the IT Systems Development Course

Topics on project management have been accommodated into the IT Systems Development course to overcome the relative lack of experience of IS students in many aspects of project management. It is to be noted that the ZU IS curricula does include a separate project management course.

Table 5 shows the *partial* mapping of several IS2002.10 topics description into our IT Systems Development course.

Table 5. Mapping of IS2002.10 Topics description into IT Systems Development Course

Topics in IS2002.10	**Topics covered in IS Software Engineering Course and project work**
Managing the life cycle: Requirements determination, design, implementation, system and database integration issues	2.6, 2.7 , 2.8, 2.9, 3.1.1, 3.2.2, 3.3.1, 3.4.2, 3.5.1, 3.5.2, 3.5.3, 3.6, 4
Network management	None
Project tracking, metrics, and system performance evaluation	2.4, 2.5, 3.1.1, 3.6, 4
Managing expectation of managers, clients, team members, and others	2.4, 2.5, 3.5.3, 4
Determining skill requirements and staffing	2.4, 2.5, 4
Cost-effective analysis	3.1.3, 4
Reporting and presentation techniques	3.5.2, 4
Management of behavioral and technical aspects of the project	2.4, 3.3.1, 3.5.3, 4
Change management	2.5, 4
Software tools for project tracking and monitoring	2.4, 3.5.3, 4
Team collaboration techniques and tools	2.4, 2.5, 4

Evaluation of the New Approach

To understand the impact of the implementation of the new approach, we conducted a survey on students who completed the IT Systems Development course and the internship course at ZU (109 students, Table 6).It is to be noted that the small size of the sample is due to the small sizes of classes at ZU. The survey was conducted by filling a hardcopy questionnaire. Table 6 shows the questionnaire and results.

For students who took the IT Systems Development course *before* the internship course: We asked students' opinion to learn about if the course helped students to develop their ability to communicate clearly with team members at their workplaces. 87% of students responded positively to this question. All students indicated that the course helped them work as part of a software development team. 85% of students responded positively to the question "the course helped me learn how to become a team player, how to schedule work between team members, and how to cope with time pressure and hard deadlines". All students indicated that the course helped them deal with non-technical issues of software projects. It also helped them (99%) learn "best practices" in SE profession. Finally, it was interesting to know that 95% of students felt that the course helped them overcome their lack of experience in many aspects of project management and software development.

For students who took the course *after* the internship course: To learn about students' opinion if they would have felt more confident and more productive during their internship if they took the IT Systems Development course before doing the internship. Results show that 70% of students replied positively while 30% said it would have made no difference. Furthermore, 98% of students agreed with the statement "The key skills I learned in this course would have helped me during my internship".

We believe the above improvements on students learning would not have been possible without accommodating topics on project management and SE best practices using Rational Suite Enterprise Tools into IT Systems Development course.

Table 6. Survey

Accommodating Topics on Project Management and Software Engineering Best Practices Lab using Rational CASE into an IT Systems Development Course:				
Survey Question	**Strongly Agree**	**Agree**	**Disagree**	**Strongly disagree**
Helped me develop the ability to communicate clearly with team members.	20%	67%	13%	0%
Helped me work as part of a software development team.	50%	50%	0%	0%
Helped me learn how to become a team player, how to schedule work between team members, and how to cope with time pressures and hard deadlines.	10%	75%	12%	3%
Helped me deal with non-technical issues of software projects.	22%	78%	0%	0%
Helped me learn "best practices" in software engineering profession.	20%	79%	1%	0%
Helped me overcome my inexperience of project management and software development.	25%	70%	5%	0%
I would have felt more confident and more productive during my internship if I took this course **before** doing the internship.	50%	20%	30%	0%
The key skills I learned in this course would have helped me during my internship.	62%	36%	2%	0%

Conclusions

The IT Systems Development course developed by the author is meant to provide an avenue for balancing the organizational context of IS with the technology context of CS. Survey shows that accommodating topics on project management and software engineering best practices lab into an IT Systems Development course strengthened students' lack of experience in many aspects of project management, and strengthened their SE practices. It also helped students to develop their ability to communicate clearly with team members in their workplaces. We believe, this finding should be propagated to other IS courses to improve students' communication skills by allowing them to work more in teams than in individual projects.

References

Abelson, H., & Greenspun, P. (2008). *Teaching software engineering-lessons from MIT.* Retrieved Oct. 2008 from http://philip.greenspun.com/teaching/teaching-software-engineering

Academic Program Model. (2008). *Zayed University Academic Program Model*. Retrieved Oct. 2008 from http://www.zu.ac.ae

Bach, J. (1997). SE education: We're on our own. *IEEE Software, 6*, 26-28.

Burns, J., & Robertson, E. (1987). Two complementary course sequences on the design and implementation of software products. *IEEE Transactions on Software Engineering (TSE), 13*(11), 1170-1175.

Cohen, P. (1987, March). The education of the information systems engineer. *Electronics & Power*, pp. 203-205.

Colvin, J. (2005). Teaching software development to non-software engineering students. *Proceedings of Conf. on Frontiers in Education: CS and CE (FECS'05)* Las Vegas, Nevada, USA, June 20-23, pp. 3-9.

Diaz-Herrera, J., & Powell, J. (1998). Educating industrial-strength software engineers. *Proceedings of the Eleventh Conference on Software Engineering Education and Training, IEEE Computer Society*. pp. 139-150.

Garratt, P., & Edmunds, G. (1988). Teaching software engineering at university. *Information and Software Technology, 30*(1), 5-11.

Gorgone, J. T., Davis, G. B., Valacich, J. S., Topi, H., Feinstein, D. L., & Longenecker, H. E., Jr. (2002). *IS2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems*. Retrieved Oct. 2006 from www.acm.org/education/is2002.pdf

Hilburn, T., Bagert, D., Mengel, S., & Oexmann, D. (1998). Software engineering across computing curricula. *ITiCSE '98, Dublin, Ireland*. Retrieved Sept. 2008 from http://www.dimap.ufrn.br/~cccc/reforma/se_curr.pdf

Keren, G. (2008). Why do universities fail teaching software engineering? Retrieved Apr. 2008 from http://users.actcom.co.il/choo/lupg/essays/software-engineering-and-uni.html

Kerzner, H. (2003). *Project management: A systems approach to planning, scheduling, and controlling* (8th ed.). Wiley.

Rational Suite Enterprise. (2008). Retrieved Oct. 2008 from http://www.rational.com

Sherman, M., & Drysdale, R. (1988). Teaching software engineering in a workstation environment. *IEEE Software, 3*(May), 68-76.

Summerville, I. (2006). *Software engineering* (7th ed.). Addison Wesley.

Shaw, M. (2000). Software engineering education: A roadmap. In A. Finkelstein (Ed.), *The future of software engineering* (pp. 373-380). ACM.

Umphress, D., Hendrix, T., & Cross, J. (2002). Software process in the classroom: The capstone project experience. *IEEESoftware, 5* (Sep/Oct), 78-85.

Weaver, P. (2004). *Success in your project.* Prentice Hall.

Biography

Abdallah Tubaishat is an associate professor in the College of Information Technology at Zayed University, United Arab Emirates. He received his PhD degree in Software engineering from Illinois Institute of Technology, USA in 1994. Dr. Tubaishat has around fourteen years of experience in teaching and research. His research interests include e-learning and software engineering. He has published a book with others entitled "Computer Skills", and has around twenty three Journal and conference publications.