

# THE TESTING OF AN EXPERIENTIAL FRAMEWORK FOR TEACHING SYSTEM DEVELOPMENT PROJECTS USING A COLLABORATIVE PROJECT MANAGEMENT APPROACH

**A. James Wynne**

Virginia Commonwealth University  
ajwynne@vcu.edu

**Randall G. Sleeth**

Virginia Commonwealth University  
rsleeth@vcu.edu

## Abstract

*Experiential learning approaches have been especially effective in information systems courses where 'real-world' scenarios are used to provide a degree of complexity that reflects actual system development projects. Students are shown how the principles of project management can provide a framework for reducing the difficulty and complexity of the development process to a more manageable procedure as they put into practice their knowledge and skills from previous coursework in creating working application systems. This paper outlines an experiential approach for teaching the value of information systems project management in the systems development process through the collaborative efforts of the graduate class in Information Technology Project Management, and the undergraduate senior capstone projects course.*

**Keywords:** experiential learning, project management, systems analysis, systems design, Unified Process, project teams, pedagogical framework,

## A Pedagogical Framework: Rationale and Benefits

Learning how to build information systems requires students to perform those activities that lead to a functioning application. Too often students are taught only to document the conceptual features and functionality the final application will incorporate. Anecdotal observations from years of teaching systems analysis, design, and implementation indicate graduates as new hires lack the confidence and know-how to make a real contribution in a real system development project. Employers are seeking to hire students who exhibit not only a conceptual understanding of systems development, but the practical experience of building a working model. Traditionally, information systems students tend to miss out on the "nuances" that are difficult to teach in a classroom environment. These nuances are the real-world experiences that collectively contribute to the competency of information technology professionals. Most classroom systems development projects fall short in offering a reasonable challenge and fail to create an engaging environment for the students to build their level of knowledge and skill. Learning becomes piecemeal, disjointed and boring, as students see only simple dimensions of complexity for the total problem domain. Lacking sufficient challenge, students fail to assume responsibility and ownership in their projects (Wynne, et al, 2005).

This paper describes pedagogy for teaching the systems project course using a collaborative interaction between (1) an undergraduate systems development project course and (2) a graduate course in information technology project management. This approach represents part of an experiential framework that is designed to enhance student learning the analysis, design and implementation of information systems. Combining systems design and the

pragmatic management tools and techniques of project management synthesizes a full system development project and depicts a realistic business information system. This approach delivers learning value for students who experience and thereby better understand the importance of the triple constraint of project management—scope, schedule, and budget, within the context of system quality and customer satisfaction. Whereas the learning objectives for the undergraduates include demonstrating skills and knowledge by building a working prototype of an information system in a project managed environment. Learning objectives include (1) understanding the behavioral issues in managing IT projects, (2) experiencing how IT project management techniques increase project success by meeting the work plan schedule, controlling scope creep, and staying within budget.

## ***Background***

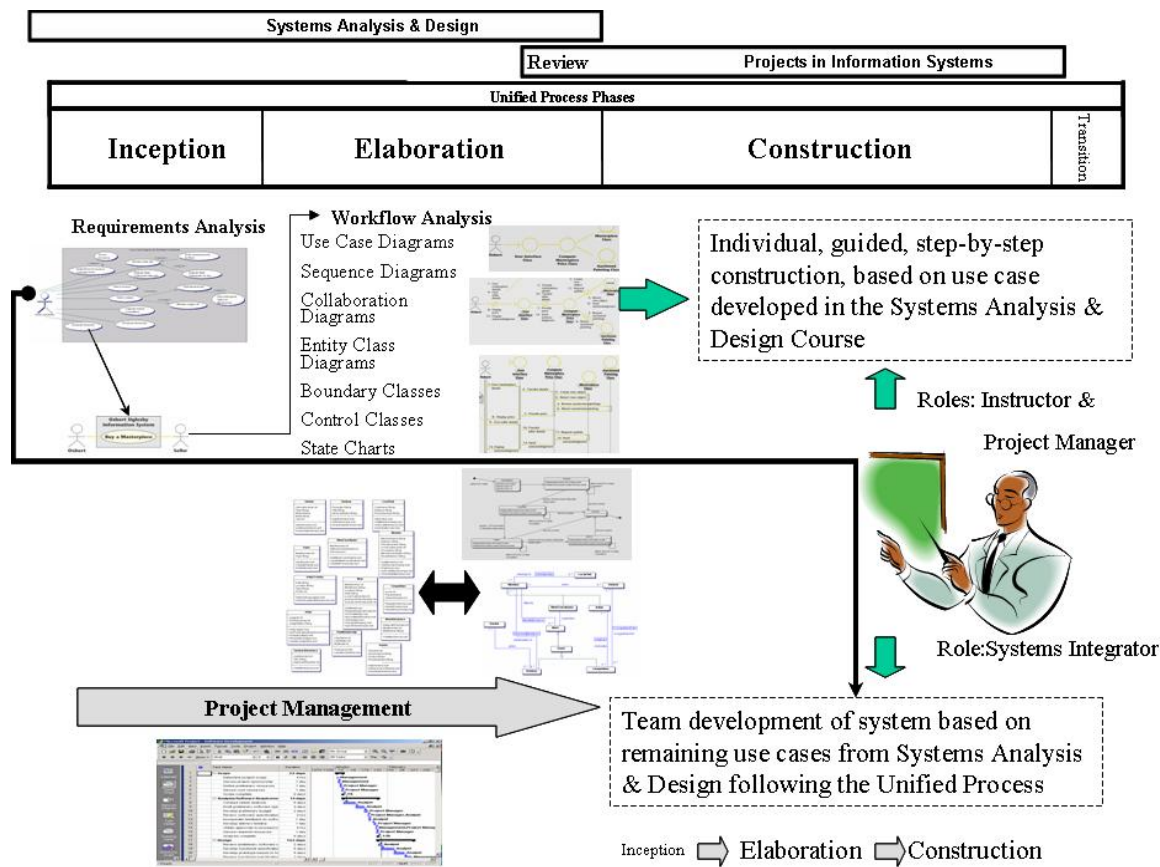
While learning information systems (IS) development, students draw upon skills previously acquired in the core curriculum courses where students learn computer hardware and software systems, programming, systems analysis and design, database systems, and data communications, and requires interrelating and integrating these skills to create new systems. Successful students also must plan, organize, monitor, and control development activities while employing management techniques, skills, and knowledge. Thus, we argue that the knowledge base of these future IS developers should include the practice of project management, and modern pedagogy in information systems development should not only focus on the integration of the IS core areas, but also on integrating them with formal project management techniques.

The basis for this approach is an undergraduate two-course sequence used to teach students the principles of the Unified Process, an Object Oriented Systems Analysis and Design methodology [Jacobson, 1999]. The first course in the sequence, Systems Analysis and Design, focuses on requirements, analysis and design workflows of the Inception and Elaboration Phases; whereas the second course in the sequence is the capstone or “Projects” course, which concentrates on the design, implementation and test workflows in the latter part of the Elaboration phase, and the Construction phase (see Figure 1).

## **The Projects Course Structure**

The capstone course (or Projects course) directly builds on, and ideally should immediately follow, the Systems Analysis and Design course (SAD) (ideally, the Projects course should directly follow the SAD Course). A major component of students’ performance in the SAD course relies on the ability to master notation (e.g. UML), process (e.g. Unified Process), and a CASE tool (e.g. Rational Rose). Students’ assignments in that course apply UML modeling skills to represent the system requirements derived from the chosen business case scenario, and from discussions with the instructor. These models and the supporting documentation are the basis for the *system proposal* that students complete at the end of that course. This proposal includes requirements definitions, economic, organizational, and technical feasibility analyses, and Use Case, Sequence, Class, State Chart, and Activity diagrams depicting the results of the analysis and design iterations of the Inception and Elaboration Phases. In addition, it includes a project management work plan with a detailed work break-down structure, estimated durations, and resource allocations. A project planned budget and a project baseline flows from resource allocation decisions. The initial project plan reflects the semester calendar, identifying holidays, semester breaks, and test days.

The Projects course consists of three parts. Part 1 serves to acquaint students with the development environment, reviewing needed technical skills and business knowledge that the students should have acquired in their previous information systems and business core courses. In Part 2, students build the components of the system as specified in the system proposal they developed in the Systems Analysis and Design course. Part 3 challenges students to integrate components into a functioning prototype. We elaborate in the sections that follow.



**Figure 1. The Unified Process Phases and Workflows within the Project Management Framework (Wynne et al.)**

### ***Part 1: Ensuring the Prerequisite Skills***

In Part 1, lectures and in-class demonstrations provide reviews of some of the skills necessary to complete the application as prescribed by the chosen business case (and as given in the system proposal). These skills include programming, database development, interface design, UML modeling, project management, and some basic business processes. To demonstrate mastery of these skills, students must complete a small project; for example, develop a small application based on one of the use cases from the *system proposal*. All students work alone on the same project, using sample scripts posted on the class website and discussed in class. The project may include software to authenticate users and allow those with sufficient privileges to perform basic table updates (as required in most applications). Having students do an individual project also allows individual assessment of student performance. At the end of Part 1, students will demonstrate their individual projects in a lab setting.

### ***Part 2: Developing the Use Cases***

For part 2, students form small teams assigned to different use cases from the system proposal they developed the previous semester. After in-class discussion of the use case diagram from the system proposal, student teams receive assignments to a set of the most important functions (i.e., Use Cases). These Use Cases constitute the scope of the work that will be identified in the hierarchy of activities, task, and milestones in the project management work breakdown structure (WBS). In a small class section, more than one team may work on the same use case. The instructor functions as the program manager and keeps track of the separate project plans of the individual student teams. By design, students will work in pairs, completing all parts of their subprojects together, as in pair programming [2]. Regular class meetings provide progress reports to ensure that all teams stay on track and opportunity for the use of common database tables. The instructor also functions as the database manager for Part 2 of the course and Part 2 will encompass one-third to one-half of the total course time. Part 2 includes several iterations of the implementation and testing workflows toward the end of the Elaboration phase, and throughout the Construction phase of the Unified Process. At the end of part 2, students submit detailed documentation of the units they developed and demonstrate their units in a lab setting.

### Part 3: System Integration

In part 3 of the course, the original teams dissolve, and new teams form, based on the Use Cases they wish to develop in the integration of the prototype modules. In this last part (less than one third of the total course time), students integrate into a functioning application the individual components developed and unit tested in part 2. By the end of part 3, students produce a functioning prototype version (alpha version) of the application that must satisfy the requirements of the system proposal. Just after this point in the Unified Process, the project would enter the Transition phase. Table 1 summarizes the three-part structure of the System Project course.

**Table 1. Summary of Systems Project Course Structure**

	<b>Objective</b>	<b>Activity</b>	<b>Student Role</b>
Part 1	Ensure requisite skills and knowledge	Complete a small project representing common system functionality (one use case from system proposal)	<b>Comprehensive</b> (work individually on whole project)
Part 2	Experience all aspects of developing a system in a small team	Complete a larger project, broken down by functional areas (remaining use cases from system proposal)	<b>Shared</b> (work in small teams or pairs designing, coding, and testing components from assigned use cases)
Part 3	Experience a specialized role in a project team	Integrate the components developed in part 2 into a cohesive and functioning application	<b>Specific Function</b> (work in larger teams, depending on class size, in defined and limited functional roles)

### Tools

Students select their tools from several options. Through IBM's SEED program students have access to Rational Rose and the new XDE (Extensible Development Environment). The XDE provides some 'round trip' engineering capabilities for database and .NET projects. Further, Microsoft's 'Academic Alliance' provides their software for students enrolled in our program. .NET is available supporting C# and Visual Basic. Rational Rose is used as the CASE tool for UML modeling and documentation. Microsoft Project is used for project management. Windows XP Pro and Server 2003 may also be downloaded. Oracle is available on a departmental server for students who want to continue with these tools which were used in their database course. Enterprise Architect is available in our department's labs. In addition, Open Source tools are freely available. The instructor operates a Linux-based web server running Apache, MySQL, and PHP.

### Course Collaboration Framework

During the first three weeks of the semester, the undergraduate System Project students start Part 1 activities and tasks, and the graduate Information Technology Project Management students focus on the basic principles of IT Project Management. The graduate class exposes students to the nine knowledge areas from the Project Management Institute as a framework for initiating, planning, executing and controlling, and closing out a project (PMI, 2004). These nine knowledge areas include four **Core** Knowledge areas, four Facilitating Knowledge areas, and one Integrating Knowledge area. In addition, the graduate course introduces students to (1) Microsoft Project as a project management tool to support the facilitating knowledge areas, and (2) project management techniques such as managing along the critical path, critical chain management, and earned value analysis. Students complete small, individual assignments in three-person groups to re-enforce the concepts and to become familiar with MS Project.

We initially use Blackboard® as a groupware tool for intranet communications and file sharing among the team members. We have found Blackboard limited in inter-class support of multiple teams from different classes – a functionality we seek in a matrix or project-oriented organization. Therefore, we use Google® Groups to facilitate the organization of both (1) multiple intra-course project teams and (2) a series of inter-course project development teams. The instructor creates the Google®Group and invites the graduate students to join. A project charter sets

the protocols and standards for communication between the various teams. A communications plan standardizes naming conventions for versioning the work plan during the planning phase.

We task the teams with creating an initial WBS for each of the functional use cases they have been assigned, from which the master work breakdown structure will be developed. Undergraduate student teams first perform a series of iterations that require the System Project students to develop more detail in each of the assigned use cases. Using the results from these iterations, the graduate students work together as individual subproject leaders. The instructor for the undergraduate System Project course defines the durations for the tasks in terms that seem to conform to what students associate with typical tasks that they must complete in other classes. Typically, the duration time unit for most tasks is “about two hours.” With no prior experience in determining a more accurate measure of true time for the task(s), the initial timeline builds from these initial estimates. The importance of accurately defining the unit of time (length of time – or optimal chunk of time) is the key to think in hours vs. days, or to think in more-precise numbers of hours. This is the first opportunity that students experience the difficulty in estimating task durations, and the accuracy of durations becomes more evident as the project progresses and conflicting projects (other course work) delay the completion of some tasks. This process becomes one of several scheduling challenges that the graduate students face when establishing an accurate timeline. In many cases, students wait until the last minute to begin their assigned task, and then work “overtime” to get it completed on schedule. They learn that ‘overtime’ is not rewarded because the original schedule did not include overtime.

## Student Assessment

About 30% of a student’s grade is determined by individual performance on projects assigned in Part 1, which is assessed in multiple parts. Students first submit a project plan, and later, system documentation as well as prototypes that are demonstrated in class. About 40% of the grade is determined by the group project in Part 2, which is also assessed at various points during the course, in the form of an updated proposal, documentation and demonstration of the components developed in Part 2; and the final application and active involvement from Part 3 of the course.

Earned Value Analysis is used to determine the level of project value during the course of the project. From the initial planned value that was determined as the project progress during the semester. As the graduate students record the actual times of the deliverables, an actual cost is calculated and the earned value is posted graphically for each team to see. Both graduate students and undergraduate students can then better understand the misconceptions that the typical triple constraint measurements for determining the success of a project portray. The project development team may have met the milestone delivery date, but they used more (overt) time to achieve that result, thus experiencing a higher cost than what was in the planned budget. The result is a negative earned value, which is a more accurate measure of the projects progress that whose impact must be addressed by the project managers, e.g., the need to monitor and control the project activities. Instead of memorizing a list of benefits derived from EVA, students experientially reach the many of the same conclusions discovered by industry experience in using project management techniques for better management of projects. Earned Value Analysis is an industry standard way to:

- measure a project’s progress,
- forecast its completion date and final cost, and
- provide schedule and budget variances along the way.
- provide consistent metrics, and
- help evaluate and compare projects.

EVA enables the project management teams to integrate the three measures of the triple constraint of:

- Cost
- Schedule
- Work accomplished

The final 30% of each student’s grade depends upon mid-term and final exams, which assess knowledge and skills using traditional objective and written formats. These tests help ensure that students individually prepare to contribute to their teams. While the knowledge aids project work, the project work then aids understanding and improves exam performance.

As professors, we know that projects often fail from the lack of communication among the project manager and project team members. And, we know that project management techniques and tools can help provide the substance and timeliness needed to create support the communications necessary to manage projects effectively. The approach we

describe here introduces both project team managers and project developers to situations requiring difficult decisions to ensure acceptable performance and to encourage improved performance. The linked undergraduate and graduate classes together experience these difficulties, just as in real-world projects. Iteratively, the processes and decisions generate improved communications and better task outcomes in terms of the time, budget, and scope initially set.

## Conclusions

The observed major strengths of the framework include:

- Students gain exposure to real-world business domains
- Students gain experience with the significance of project management in the systems development process
- Students learn Project Management skills and tools
- Students integrate programming, database management, data communications, and hardware/software skills
- Students experience the complete Systems Development Life Cycle in a project management framework
- Students integrate real business process concepts.

This framework has so far been partially implemented and is continually evolving. We observed that the ability for students to learn in an “engineered project management” real-world environment has provided more integrity to the learning process and assessment. Establishing performance expectations that challenge students in a positive way has helped develop their individual sense of competence and accomplishment.

## References

- [1] Jacobson, I., Booch, G., & J. Rumbaugh, *The Unified Software Development Process*. Addison Wesley, 1999.
- [2] Lowell, L., and R. Jeffries (2004). Extreme Programming and Agile Software Development Methodologies. *Information Systems Management* 21(3), 41- 52.
- [3] Wynne, A. J., H. R. Weistroffer, and G. Saunders, *Proceedings of the 2006 Decision Sciences Institute*, San Francisco, CA, 2006.
- [4] PMI, A Guide to the Project Management Body of Knowledge, *PMBOK® Guide*, The Project Management Institute, 2004.