Data Engineering Education with Real-World Projects

Paul S Grisham, Herb Krasner, and Dewayne E. Perry

Empirical Software Engineering Lab (ESEL)

ECE, The University of Texas at Austin

{grisham, hkrasner, perry}@ece.utexas.edu

Abstract
This paper presents an experience report on teaching Data Engineering using a real-world project domain. Our course introduces databases within the context of Systems and Information Engineering, supplementing relational database theory with requirements engineering, design, and analysis. The primary deliverable of the course was a semester-long project to implement an information system in a real-world application domain, interacting with an external customer with uncertain requirements. We believe that real-world projects motivate students to apply good Software Engineering principles in the classroom and encourage those principles to be adopted into industrial practice.

Keywords
Software Engineering Education, Systems Engineering, Requirements Engineering, Database Systems.

1. Introduction
The Center for Lifelong Engineering Education (CLEE) at the University of Texas at Austin offers a Master of Science degree program in Software Engineering in Engineering for practicing professionals. This program, commonly referred to as Option III, is organized to accommodate a full-time work schedule. Classes are intensive, and meet one weekend per month. The program of study includes 33 graduate credit hours, and is a combination of standard classroom lecture, topical conference courses, and a Master's Report. In the Fall Semester of 2004, a new course, *ECE 382V: Data Engineering*, was added to the curriculum. The course was designed as an introductory course on database concepts within a Systems and Information Engineering context.

The use of real-world projects in software engineering or requirements engineering classes is not new, but in designing the new course, we wanted to motivate the connection between relational database theory, design, and the practical software engineering aspects of systems and information engineering. To accomplish this goal, we used a single, semester-long, real-world project – a course registration system with complex and ambiguous requirements – that each student group would design and implement. The project environment relied on the availability of a customer representative who provided requirements and evaluated the final deliverables. We believe that the experience of applying the theoretical aspects of relational databases to this project in a dynamic environment motivates the adoption of good Software Engineering principles outside the classroom.

2. The Project Domain
The CLEE Online Registration System is a web-based application to provide a management system for CLEE's various courses, conferences, and training programs. Instead of designing and implementing a generic class registration system with generic requirements, we selected the CLEE domain specifically because it had complex requirements, such as state regulations for certification reporting and data integration with other university information systems.

Without the system, CLEE staff must manually process each registration, as well as process invoicing, billing, and payment. Moreover, the University of Texas system utilizes several large information systems for tracking financial, auditing, educational, logistical, and licensing and certification information. CLEE staff must enter the event and registration information into these systems manually.

The goals of the CLEE Online Registration System project are to automate the existing portions of the existing online registration system and to add new functionality for logistic support. The new registration system must integrate registration with course management and maintain historical records for use by individual users and CLEE staff. The system must also generate a variety of financial and marketing reports.

3. Course Organization
The classroom component of the course consisted of ten, four-hour lectures, meeting approximately every fourth weekend from August 20, 2004, through December 3, 2004. Lectures were highly intensive, with reinforcing homework and advance readings assigned between class meetings. A take-home mid-term examination was also assigned to test student understanding of relational database theory.

The class project was designed to run for the duration of the semester. The class was divided into ten teams of four or five students. Early homework assignments required students to perform requirements analysis, while later assignments called for students to create designs and schemas that formed the basis of their project implementation.

Students began by evaluating and modeling the current system. Over the course of the semester, group interviews were conducted with a customer representative during class time. The class used Blackboard, email, and other online coordination tools to share information about requirements. During the long period between classes, questions about requirements were emailed to a member of the teaching staff, who compiled them, interviewed the customer representative, and made the answers available to the class.

Based on the requirements provided by the customer representative, a comprehensive System Requirements Specification (SRS) document was compiled by the teaching staff and delivered to the students. The structure of the SRS was derived from several sources, and the final SRS is outlined in Figure 1. By this point in the semester, students had already submitted their initial data models and schema based on the unorganized requirements. With the SRS, students were asked to review the requirements and revise their designs, identify shortcomings, and clarify still uncertain requirements.

The SRS defined 13 operational scenarios, 40 functional and data requirements, and 16 non-functional requirements. Students were required to design their data models and schema to accommodate all of these requirements. For purposes of implementation, the scope of the project was scaled down to require implementation of only 10 scenarios, 39 of the functional requirements, and 9 of the non-functional requirements. Many of the eliminated requirements were technology requirements outside the scope of the class.

1. Introduction
   - Purpose, scope, definitions, acronyms, etc.
2. General Description
   - Existing system analysis
   - Stakeholders
   - Goals
3. Operational Scenarios
4. Functional and Data Requirements
5. Non-Functional Requirements
   - User-Interface Requirements
   - Software Interface Requirements
   - Performance Requirements
   - Security Requirements
   - Class-Specific Requirements
6. Open Issues
7. Delivery Requirements and Schedule

Appendix: Collection of CLEE data artifacts

Figure 1. SRS Organization

In the interest of making the project fair for all students and streamlining the technical support, students were required to implement the project using open-source tools for their database, application server and programming language. Students were required to hand-code their databases using SQL, and were not allowed to use advanced modeling tools that support automatic generation of code. Students were allowed to use any coordination, groupware, or version management tools they chose.

4. Project Evaluation

Project evaluation was divided into four major components: homework, project notebook, project demonstration, and peer evaluation.

The homework assignments represented incremental delivery of the project, based on the current state of the project. The initial homework assignment, for instance, was to review the current state of the CLEE Online Registration System, perform an initial evaluation of data requirements, and generate a set of questions for the customer. As the lectures covered more database theory, students were required to generate models and schemas, revising their designs as the requirements evolved. We attempted to provide as much constructive feedback as possible for each submission and revision.

The project notebook represented the final version of the various models and schemas generated and revised throughout the semester, including the data design as well as the implementation sources and any project analysis provided by the students. Specifically, the project notebook was required to contain:

- The E/R model of the problem domain
- A data dictionary of the problem domain, the relational schema of the database
- The SQL DDL of the database
- The SQL DML for all queries used by the required scenarios
- Complete source code (PHP, HTML, CSS, and graphics used by the website)
- Test cases and results
- A brief discussion of the design challenges and compromises the team encountered
- A brief evaluation of the technology used for the project

The project notebook was manually inspected to ensure that each scenario and requirement within the project scope was sufficiently implemented. Grading used a spreadsheet-based instrument to track coverage of requirements.

A significant portion of the final notebook grade was reserved for analysis and discussion. Students were instructed to provide a "brief, but insightful" discussion of the major design challenges and compromises as well technical evaluation of the software tools used in the class. We were intentionally vague on this requirement, and many of the groups impressed us with their level of critical analysis of their own projects.

Each group was required to demonstrate their project in class to an evaluation panel made up of the customer representative, and teaching staff. Time did not permit a full acceptance test, but instead, each group was assigned a unique subset of scenarios to demonstrate. The evaluation panel attempted to select an mix of simple and complex scenarios and to exercise distinct areas of the implementation. Each team was also required to demonstrate one scenario of their choosing, which offered a chance to explain some especially innovative or interesting aspect of their implementation.

The evaluation panel used a standard instrument for grading, which measured each team's performance in a number of qualities against a Likert scale. The questions used in the demonstration evaluation instrument are listed in Figure 2.

The final component of the project grade was peer evaluation. Peer assessment offers many benefits, and is gaining in acceptance as a necessary element in the classroom. During the demonstrations, students were required to observe the demonstrations and provide feedback. Each student was provided a simplified version of the demonstration evaluation instrument.

In addition, group members were required to rate their teammates in terms of contribution and level of effort, as well as their own performance on the team project against that of their teammates. This gave the teaching staff the ability to evaluate group performance in the presence of extenuating circumstances or troublesome team members.

The final projects were generally satisfactory, and occasionally exceptional, and the grading reflected the generally high quality of the students' work. Final course grades were commiserate with expectations for highly motivated graduate students. (avg.: 92.5; med.: 92.9; std. dev.: 4.50, where 90-100 is an A)

The team seemed prepared and confident.

The team answered all my questions satisfactorily.

The team website seemed easy to use.

The team website was visually appealing.

I was satisfied with scenario demonstration X.

My overall satisfaction with the system as demonstrated was…

Figure 2. Demonstration Evaluation Questions

We were concerned that the uncertainty in the project would overwhelm our busy students, and lead to frustration and resistance. Instead, the project grades were generally higher than the overall course grades. (avg.: 94.7; med.: 95.2; std. dev.: 4.67) We found that, overall, the students coped with the complexity and uncertainty.

The exam scores were generally lower than the overall class grades. (avg.: 87.0; med.: 89.0; std. dev.: 9.00, which includes a 3 point positive curve) These numbers suggests that the project component actually improved the overall class grade. For dedicated team members on dysfunctional teams that produced unsatisfactory projects, excellent exam, homework, and individual participation scores could be sufficient to merit a higher grade.

5. Discussion

5.1 Requirements Uncertainty

The distinguishing characteristic of our project is the complex and uncertain nature of the requirements the students were dealing with. The project was sufficiently complex that there was no single correct solution to the problem. Moreover, it was clear early in the semester that a full implementation of the system was impossible within a single semester.

Requirements uncertainty can derive from instability (changes over time) and diversity (the differences in understanding between stakeholders) [5]. We also consider poorly understood requirements, those that are incomplete or ambiguous at the point of design [6]. In this project, uncertainty arose from the diversity of stakeholder requirements and incomplete requirements that were exposed through negotiations between the students and the customer proxy.

We intentionally allowed requirements to remain vague and uncertain for as long as the students left them. Students recognized that it was their responsibility to clarify and resolve requirements uncertainty. The final, official version of the SRS, including implementation scope, was not given to the students until approximately 3 weeks before the project demonstrations. At this point, the students were expected to have completed their working relational database design and be implementing it.

Despite their best attempts to clarify the requirements, there were several requirements the students never adequately understood. For instance, eight of ten groups failed to deliver a satisfactory implementation of the marketing effectiveness report as defined by the SRS. The customer provided example marketing reports and sample operational data, and discussed the marketing reports requirement at every customer interview. We found that there was a disconnect between the way that the customer expressed the problem as a feature and the way the student's thinking on the problem as designers.

We did not intentionally add complexity or confusion into our process but allowed the situations to develop the way that they do in real projects, only rarely becoming involved to arbitrate conflict. We differentiate our real-world problem approach from the controlled failure environments of the Live-Through Case Histories approach, which intentionally inject failure scenarios into an ongoing class activity [3].

The teaching staff had to constantly monitor the project and adjust the scope and requirements to bring the final project expectations to an appropriate level of effort. At several points during the semester, we had to reassure the students that the final project scope would be manageable by their project teams, assuming that they had stayed current with the deliverables. We had to be willing, even at the last minute, to scale down the project if we perceived that we had misjudged the level of effort or skills of our students.

It is exactly for this reason that we think that this type of project can be used in other types of classes. Even though our students were mature, highly motivated, and often had years of technical experience, we found that the infrequent class schedule made incremental delivery and immediate feedback difficult. With a traditional graduate class, the instructor and students interact two or three times per week, instead of twice per month.

5.2 Student Teams

Originally, we planned to assign students to the project teams, but after resistance from the students, we allowed the students to form their own teams. The basis of the students' concerns were that many of the students had worked together on teams in the past, and they already knew how to overcome the differences in geography, work schedule, etc. Since many of our students were from out of town, it seemed reasonable to allow them to form teams that would minimize coordination difficulties. Prior work suggests that successful teams need time and face-to-face collaboration to build trust and agree on team goals [1].

In practice, it worked extremely well for some teams but was maximally inconvenient for other groups. One group could be formed of four database technologists who work for the same company in the same city, while people who live and work in different states and didn't naturally join with another group could form another group of teammates.

In allowing teams to self-form, we also did not consider the technical expertise of the groups. We could have tried to balance teams with respect to the level of experience students had. Some of our students were practicing database technologists, while others had no prior formal computer science or engineering education. Studies of student teams suggest that students respond well to teams in which they perceive that their partners' skills are comparable to their own [2], and it is reasonable to assume that this perception was also a motivating strategy in students' team formation.

There are many methods that can be used to build fair teams in the classroom [4]. In industry, project teams made of members of equivalent experience and capabilities are rare. There is a positive impact on student perceptions and performance working when working with familiar teams, but there is also a benefit in approximating team situations that motivate the need for process and team management. As the goal of the project is to teach good software engineering practices, the teaching staff must create an environment that allows motivated students to succeed even in the context of a dysfunctional team.

5.3 Technologies

Our customer actually preferred that we use the same commercial web application system that the university's IT group uses, but we could not afford to provide those tools to our students. We admit that in this case, our decision of development platform was somewhat arbitrary.

We asked the students to provide a technology evaluation as a part of their project notebook submission. Student responses varied from simple claim that the technologies used in the class were sufficient for the project, to extensive comparisons with other technology options.

Students used additional technologies of their choice for implementing and managing their project, such as version management, programming editors, and code libraries. We encouraged our students to provide technical evaluation for all of these tools. For many of our students, even those with a background in databases, we found that the course project provided them with a useful experience in technology evaluation.

5.4 Intellectual Property Concerns

Students were generally concerned about the level of effort required to complete the project and grading standards. However, we heard a very common concern from our students that we found surprising. Our students were very concerned that we would take their projects and deploy them without properly compensating them for their work. They recognized that the class activities were very similar to their own real-world jobs. One student said that he dealt with very similar situations as part of his job. We viewed his comments as an indication that we were providing a good environment for applying the theory and disciplines we were teaching.

In general, though, we believe that although the requirements are real, the class project implementations should not necessarily be deployable. Performance, security, reliability, and even licensing issues are well beyond the scope of a one-semester class on data engineering. In addition, it is important to define up-front to the customer representative what their involvement will be and what the expected deliverables are [6]. In our case, the customer representative was a member of the CLEE staff, which meant that we did not have to manage expectations with an external organization.

5.5 Good Software Engineering Practices

In their project notebooks, many students talked about their experiences in trying to coordinate a large uncertain project. Our students determined quickly that risk factors like having geographically dislocated teams, uncertain requirements and fluctuating problem scope, and lack of coordination tools such as version control, would eventually undermine the success their project. We encouraged our students to develop their own best practices and discuss them with the teaching staff and with each other.

The most common issue students shared with us related to requirements uncertainty and implementation scoping issues. Several teams developed methods to arbitrate differences in requirements interpretation within their teams. Others attempted to define concrete acceptance criteria for those requirements. One team actually designed their data model for maximum flexibility to respond to the stream of changes and clarifications over the semester. That team went on to discuss the long-term cost of the flexibility in their design in terms of time and complexity.

The second most common issue for teams was team coordination and management. Several teams had team members in multiple cities, or even multiple states. We expected that students would use the team coordination facilities in Blackboard to manage documentation and design arbitration. Instead we found that many teams used undocumented meetings, phone conversations and email, relying on *ad hoc* coordination methods to manage their team. Many teams expressed regret that they had not been more disciplined in the use of standardized naming conventions, version management, and other coordination methods.

One team, however, established an organizational structure, and even named a data model coordinator. This team defined a structured process to propagate changes to requirements through the documentation, data model, and even the implementation itself. This team had comparatively few crises, and was successful at distributing the level of effort over the semester, instead of scrambling at the end to meet the deadline.

6. Conclusions

During the teaching of this course, we discovered that our students were able to experience the challenge of working with a large, complex project with uncertain requirements in a relatively low-risk environment. The project provided exposure to typical real-world software engineering problems. Moreover, this approach motivated the need for good software engineering process management and disciplined data and requirements engineering.

The course structure tied the lecture material directly to both illustrative sample problems and the ongoing project context. Homework assignments were created around project deliverables, which facilitated continual feedback to the students, and ensured that the level of effort was more evenly distributed throughout the semester. The final determination of implementation scope was deferred until late in the semester and based on the approximate level of effort the students were capable of delivering.

The project domain was complex enough, that even though the students converged on a single view of the requirements, each team's data model was unique. The use of a single project domain for the entire class is appropriate because it enables a single customer representative to serve for the whole class, and because it enables the entire class to discuss and resolve requirements uncertainty.

In summary, it was clear to us that exposure to real-world software engineering issues in the environment of the classroom motivates the appreciation and adoption of good software engineering practices. To our surprise, student performance on these real-world projects was typical for team-based projects in general. Although the projects presented special challenges, they did not adversely affect overall class performance. In addition, many students actually demonstrated better understanding of the material on the project than on the examination.

7. Acknowledgments

Our thanks to Nicole Evans, Graduate Coordinator for CLEE, for being an excellent customer representative; to Jaime Puryear, Senior Marketing Manager of CLEE, for providing operational data; and to Chris Chimera, for providing technical information. We also thank the students from the Fall, 2004, *ECE 382V: Data Engineering* class for being willing to learn and work hard, and for generally having a good sense of humor about everything.

8. References

[1] Gil, Gurgit S., Dewayne E. Perry and Lawrence G. Votta. A Case Study of Successful Geographically Separated Teamwork. In Proceedings of Software Process Improvement ’98 (SPI98), (Monte Carlo, December 1-4, 1998).

[2] Katira, Neha, Laurie Williams, and Jason Osborne. Towards Increasing the Compatibility of Student Pair Programmers. International Conference on Software Engineering 2005 (ICSE’05), (St. Louis, MO, May 15-21, 2005).

[3] Klappholz, David. and Larry Bernstein. Overcoming Aversion to Software Process through Controlled Failure. Presentation. DoD Software Technology Conference 2002 (STC 2002), (Salt Lake City, UT, May 2, 2002).

[4] Michaelsen, Larry, Arletta Bauman Knight and L. Dee Fink. *Team-Based Learning.* Stylus, Sterling, VA, 2004.

[5] Sarma R. Nidumolu. Standardization, Requirements Uncertainty, and Software Project Performance. *Information & Management* (31:3) (Dec. 1996), pp. 135-150.

[6] Perry, Dewayne E. and Carol S. Steig. Software Faults in Evolving a Large, Real-Time System: a Case Study. 4th European Software Engineering Conference (ESEC93), (Garmisch, Germany), (Sept. 1993).

[7] Williams, Judith C. (Moderator), Bettina Blair, Jürgen Börstler, Timothy C. Lethbridge, and Ken Surendran. Client Sponsored Projects in Software Engineering Courses. 34th SIGCSE Technical Symposium on Computer Science Education, (Reno, NV), (Feb. 19-23, 2003).