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 as a graduate-level class using a real-world project domain. Traditional computer science database courses focus on relational database theory and typically offer a background in SQL and database implementation. Our course presented databases within the context of Systems and Information Engineering, supplementing traditional relational database theory with a strong sequence of requirements engineering, data 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 (that is, with a real, external customer with uncertain requirements in a practical business setting.) We believe that the use of such project domains motivate students to apply good Software Engineering principles in the classroom, which consequently encourages those principles to be extended into industrial practice.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education.*

General Terms

Design, Human Factors, Theory.

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 terminal[1] Master of Science degree program in Engineering for practicing professionals. This program, commonly referred to as Option III, is organized to accommodate a full-time work schedule. The classes are intensive, and only 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. The Option III program offers a concentration in Software Engineering. The courses are organized into Engineering Methods, Software Systems Technology, and Program Management.

[1] By terminal, we mean that the degree plan is appropriate for students pursuing an M.S. Degree for professional development. It is not the traditional step in the process of earning a Ph.D. We note, however, pursuit of a Ph.D. is not precluded, but must be done through the regular program at UT.

In the Fall Semester of 2004, a new course, *ECE 382V: Data Engineering*, was added to the curriculum. The course was designed to present database concepts within a Systems and Information Engineering context. The course is unique within the University of Texas at Austin. The Electrical and Computer Engineering graduate program, which has a concentration in Software Engineering, does not offer any database or data engineering courses at either the undergraduate or graduate level. The database course offered by the Department of Computer Sciences in the College of Natural Sciences is a more traditional course, focusing on relational theory and database implementations.

The remainder of this paper presents an experience report on the *ECE 382V: Data Engineering* course, and in particular, the use of a large, real-world problem domain for the course project. We believe that our experiences could help in the design of both undergraduate and graduate-level courses, which would foster the use and adoption of good Software Engineering principles in practice.

2. COURSE ORGANIZATION

The course, *ECE 382V: Data Engineering*, was initially offered in the Fall Semester, 2004. The classroom portion of the course consisted of ten, four-hour lectures, meeting approximately every fourth weekend from August 20, 2004, through December 3, 2004. Given the long time between class meetings, lectures were typically highly intensive, with reinforcing homework and advance readings assigned between class meetings. A lengthy take-home examination was scheduled so as not to interfere with the course lecture schedule or project schedule.

The class project was designed to run for the duration of the semester. The class divided itself into ten teams of four or five students. Homework assignments were closely related to the project domain and typically followed the pattern of an individual submission, followed by team discussion and a joint team submission. Early in the semester, the homework assignments required students to perform requirements analysis, while later assignments called for students to provide the designs and schemas that formed the basis of their project implementation.

The project domain, described in detail below, is an online registration system for CLEE. Instead of designing and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
International Conference on Software Engineering 2006 (ICSE 2006), May 20-28, 2006, Shanghai, China.
Copyright 2006 ACM 1-58113-000-0/00/0004…$5.00.

implementing a generic class registration system with generic requirements, we selected the CLEE domain specifically because it had complicated and uncertain requirements, such as state guidelines on certification reporting and integration with other university information systems.

Students began by evaluating and modeling the current system. Students held group interviews with a customer representative over the course of the semester. When the schedule allowed, group interviews were scheduled during class time. The class used Blackboard, email, and other online coordination tools to distribute information about requirements. During the long period between classes, questions about requirements were typically 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. By this point in the semester, students had already been required to submit and revise their data models and schema based on the unorganized requirements. With the SRS, students were asked to review the requirements and their designs, identify any shortcomings in their designs, and clarify any still uncertain requirements. The structure of the SRS was derived from several different sources, and the final organization is presented in Figure 1.

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. However, for purposes of implementation, the project was scoped down to only require implementation of 10 scenarios, 39 of the functional requirements, and 9 of the non-functional requirements. Many of the eliminated requirements were removed because they were interfacing or technology requirements outside the scope of the class. For instance, the data model must contain all information required to interface with the university's financial system, but the students need not implement the scenario that demonstrates that functionality.

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

Students were required to implement the project using Apache as the web information server, PHP4 as the programming environment, and MySQL as the database management system. In the interest of making the project fair for all students and streamlining the technical support, students were not allowed to use other databases, languages, or environments. In particular, students were not allowed to use advanced modeling tools that support automatic generation of code or SQL directives.

Grading was based on two main deliverables: the customer demonstration and the project notebook. During the final class meeting, teams were required to demonstrate their systems, this time to a panel made up of the customer representative, instructor, and teaching assistant for the class. Each panel member randomly selected a scenario, and the team was responsible to follow the scenario from beginning to end, demonstrating the specified functionality. The team was evaluated on both system performance and general preparedness to answer customer questions. The other students were also instructed to provide anonymous feedback on the team's presentation through the use of standard feedback forms.

The project notebook consisted of the revised versions of the previously submitted homework assignments, plus the full implementation and any relevant project analysis. 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 (including 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, and a brief evaluation of the technology used for the project. Source code was not evaluated for style, or even correctness. Instead, the project notebook was manually inspected to ensure that each scenario and requirement within the project scope was sufficiently implemented. The project notebook was a team submission, though each team member was also required to submit an evaluation form on the estimated level of contribution and effort put in by each teammate.

Including the homework submissions, the project comprised 70% of the total grade for the course.

3. 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.

Currently, CLEE handles event registration through several channels: the current website, faxing or mailing a registration form, or by placing an order with a CLEE staff member over the telephone. Regardless of the registration method, the CLEE staff member responsible for the event must process the registration manually. Credit card transactions, for instance, are entered by hand into a Point-of-Sale (POS) terminal. In the case of a check, money order, or purchase order, the CLEE staff member is responsible for invoicing and billing the registration and ensuring that all payment is received in a timely fashion.

Marketing data for the class are transferred from the registration forms by hand into standard business tools. This data can be used to create business and financial reports to determine which events to offer in the future. This data can also be used to create contact mailing lists and targeted advertising strategies.

The University of Texas system utilizes several large information systems for tracking financial, auditing, educational, logistical, and licensing and certification information. Depending on the type of event, CLEE staff members 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. The new system should accept registration data and begin processing payment automatically. In the event of a payment through check, money order, or purchase order, the system should provide support services for billing, invoicing and collections.

The new system must maintain historical records for use by individual users and CLEE staff. A user should be able to view a history of completed courses and certifications, as well as view and edit upcoming registrations. CLEE staff might use the historical data to track course attendance and marketing details. The system should be able to generate a variety of financial and marketing reports with different levels of detail.

4. PROJECT EVALUATION

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

4.1 Homework

The homework represented incremental delivery of the project, based on the current state of the project. The initial homework assignment, for instance, was for the students 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 requirements models, data models, entity-relationship models, schemas, and so on.

Early homework assignments were individual submissions, while later assignments were team submissions, often including revisions of previous submissions as requirements changed and solidified. We graded homework on a satisfactory/unsatisfactory scale, but we attempted to provide as much constructive feedback as possible for each submission and revision. Homework made up 20% of the final course grade.

4.2 Project Notebook

The project notebook represented the final versions of the various models and schemas generated over the semester, including up the data design as well as the implementation sources and any project analysis provided by the students. The project notebook was a team grade, and comprised 20% of the final course grade. Grading used a spreadsheet-based instrument to track coverage of requirements.

E/R Model Coverage counted for 25% of the notebook grade. The data requirements were evaluated on a satisfied/unsatisfactory/ unsatisfied basis and then summed. Model design represented a subjective evaluation of the quality of the E/R Model on the basis of elegance, maintainability, and comprehensibility. Design quality counted for 10% of the notebook grade.

The Implementation Coverage counted for 15% of the notebook grade, and the SQL DML and implementation code was similarly inspected to ensure that all scenarios and functional requirements were implemented. Scenarios were broken down into the functional requirements that implemented them.

Data dictionary, relational schema, SQL and test data made up a total of 35% of the notebook grade, and grading consisted mainly of completeness and accuracy against the final version of the data model and implementation.

The remaining 15% of the project grade was for the 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.

4.3 Project Demonstrations

On the final day of classes, each group was assigned a brief period to demonstrate their project to an evaluation panel made up of the customer representative, course instructor, and teaching assistant. The basic structure was that each member of the review panel would request one of the 10 scenarios, which the team would have to demonstrate and subsequently answer any questions that arose. Each team was also required to demonstrate one scenario of their choosing, so as to offer a chance to explain some especially innovative or interesting aspect of their implementation. In practice, the evaluation panel attempted to select an even mix of simple and complex scenarios, and to exercise distinct areas of the implementation.

Grading by the evaluation panel used a standard instrument, 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.

In addition to the evaluation panel, the other students were required to observe the demonstrations and provide feedback. Each student was provided a simplified version of the demonstration evaluation instrument. The project demonstration was a team grade, and made up 20% of the final course grade.

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

4.4 Teammate Evaluation and Participation

The final component of the project grade was participation, for which, in part, students 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. As with any group project, there were a few teams that had troublesome team members, either because they were not as capable or as dedicated as the rest of the team. A few students had extenuating personal situations that prevented them from participating fully on the project. What we found was that the students who received poor ratings by their teammates corresponded to situations of which we were previously aware.

In the end, the participation grade, which made up 10% of the final course grade, was assigned by the instructor, and was used primarily as a means of justifying special consideration for students who had been especially helpful or hard working on the project.

4.5 Course Grades

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

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. (average: 94.7; median: 95.2; std. dev.: 4.67) We found that overall, the students coped with the complexity and uncertainty and delivered generally satisfactory, and occasionally exceptional, projects.

The exam scores were generally lower than the overall class grades. (average: 87.0; median: 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 somewhat 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.

We intentionally allowed requirements to remain vague and uncertain for as long as the students left them. Some of the students recognized that it was their responsibility to clarify and resolve requirements uncertainty. The final version of the requirements, including conflict resolution and 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 designed their relational database and possibly be implementing it.

There were a few requirements that were never adequately represented to the students, no matter how the students and customer communicated about them. For instance, there was a requirement that the system be able to generate marketing effectiveness reports, that is, which marketing methods were most cost effective. In order to generate this report the system must gather information from the user about how the user found out about a registered event and CLEE in general. The system must also gather information about how much money was spent on various marketing options over a certain period of time. The customer provided example reports and sample operational data. The marketing reports requirement was discussed at every customer interview.

Despite repeated attempts to clarify and model the requirement, eight of ten groups failed to deliver a satisfactory implementation of the report as defined by the SRS. Because the impact of the single unsatisfied requirement on final project grade was negligible, we found that exposure to the difficult requirement was an appropriate example for teaching the need for detailed requirements engineering.

Although our project domain was realistically complex and large, our approach was never intended to be a controlled failure environment, like the Live-Through Case Histories approach [2]. In a live-through case history, students are presented with a simulated project and given opportunities to solve problems. At various points in the project, real-world type failure events occur, giving students an opportunity to apply sound software engineering principles to correct the failure and prevent failure in the future. In this way, good software engineering practices are motivated to the students in a controlled environment (i.e., the classroom.)

Our approach was more of a controlled chaos environment. We did not inject failure or confusion into our process, but allowed the situations to develop naturally, the way that they do in real projects. Our students determined quickly that risk factors, like having geographically dislocated teams, uncertain requirements, lack of coordination tools and version control, and fluctuating problem scope, would eventually undermine their project. We encouraged our students to develop their own best practices and discuss them with the teaching staff and with each other.

This approach required the teaching staff to constantly monitor the project and possibly 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 incremental homework 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 some resistance from the students, we allowed the teams to self-form. 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 to make sense to allow them to form teams that would minimize team 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 in the same group for the same company in the same city, while another group made up of the people who didn't naturally join with another group could wind up with teammates who live and work in entirely different states.

In addition to the geographic issue, we also did not consider the technical expertise of the groups. We should have tried to make teams fair with respect to the level of programming, database, and web development experience the team members had. In addition, we feel that the argument that prior experience with team members is not a compelling reason to form a team. Forming teams with new people offers an educational opportunity to share ideas and experiences with students from other backgrounds.

In the Option III environment, we think that the geographic distribution would encourage students to employ good software engineering principles to coordinate and overcome their lack of collocation.

In a more traditional graduate student environment, the geography issue is not a factor, as all students will be relatively available to meet and work with other students. In an undergraduate class, you may assume that the students have more or less the same level of experience. There are many methods that can be used to build fair teams in the classroom [3], though we should be sensitive as to how we explain the team assignments, especially if they are based on experience of past-performance.

5.3 Technologies

We selected the use of Apache/MySQL/PHP4 for a number of reasons. The primary reason was that we wanted all teams to be using the same tools in the interest of fairness. We specifically wanted to disallow one team from using an expensive, proprietary commercial solution with many shortcuts, that another team would not be able to access. We liked that the Apache/MySQL/PHP4 solution was both open-source and freely available for our uses. Considering that our customer preferred their solution to use the same commercially available web application system that the university's information technology group uses, we admit that our decision was somewhat arbitrary.

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

Students used additional technologies 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 Student Concerns

In typical fashion, students were concerned about the project itself, especially with regard to the requirements, scope, effort, and grading, all of which was left purposefully vague until late in the semester. However, there was a very common concern we heard from our students that we were surprised by.

Since we were building a system to satisfy real-world requirements, our students were very concerned that we would take their projects and deploy them without properly compensating them for their work. In other words, they recognized that the class (that they were paying for) was very similar to their own real-world jobs (for which they got paid.)

One student expressed concerns that he would have to return to his job the following week and deal with very similar types of situations in his office. 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. The myriad performance, security, reliability, and even licensing issues are well beyond the scope of a one-semester class designed to teach specific concepts. In addition, universities have different regulations regarding the ownership of intellectual property of student works. We suggest that students be told that their class projects will not be used in deployed applications without their expressed permission.

5.5 Good Software Engineering Practices

We end this section with comments that our students provided for us about their projects. We feel that these statements illustrate how the project motivated the adoption of good software engineering practice.

The data model is extremely flexible and this made building a usable interface for the customer very difficult. So there is a cost to flexibility – implementation cost (of time and money) and complexity cost.

Consistency in naming would have helped us quite a bit. We did not agree on a common attribute and entity naming scheme. This caused confusion while writing queries where we were using each other's tables.

It would have been helpful to have a list of good tools with common IDE features such as syntax highlighting, auto completion, online help and debugging. This would have allowed the focus on the database and not learning new languages.

Until we know more about the scope of the project I wouldn't go too far down any path…. I hope the customer is going to tell us more next time.

Our team members interpreted various requirements in different ways. This lead to discussion of the root cause of the requirement and conflict resolution regarding what direction the team would follow.

We had inadequate time to properly design and prototype our application due to constant changes being made to the requirements much like a real time project.

One suggestion that I would make to someone planning a project like this would be to add the Eclipse development platform to this combination. The current version includes a PHP plug-in that is excellent. We also made extensive use of the CVS plug-in to manage our code base. It helped tremendously.

I feel the flexibility outweighs the breakage of our model, but it also indicates that our model could benefit from further analysis.

One of the major challenges our team encountered centered around the failure to lock down requirements until a relatively late stage in the project timeline…. To cope with this situation, the team took the approach to move ahead with a complete database design, making assumptions where necessary despite incomplete and fluctuating requirements. As requirements came in and questions were answered, modifications were made to the data model as necessary…. To maintain the integrity of the data model, a master copy was maintained by one team member. As the requirements, and therefore the data model itself, evolved, proposed changes were discussed at team meetings and the design was incorporated into the documentation. By keeping the documentation up to date and making changes incrementally, the task of arriving at the final database design was distributed well over time.

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 enough exposure to very typical real-world software engineering problems and motivated the need for good software engineering process management and the disciplined application of data and requirements engineering.

The course structure tied the lecture material directly to both illustrative sample problems and complex data modeling applications. Homework assignments were created around project deliverables, which facilitated ongoing feedback to the students, and ensured that the level of effort was more evenly distributed throughout the semester. The final determination of implementation scope can be deferred until late in the semester and based on the approximate level of effort the students are 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, debate, and resolve requirements uncertainty.

In summary, it was clear to us that exposure to real-world software engineering issues in the relatively safe 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. Not only did the project not adversely affect their grades, 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 registration and marketing data; and to Chris Chimera, for providing technical information about the current state of the CLEE information system.

Of course, none of this would have been possible without the students from the Fall, 2004, *ECE 382V: Data Engineering* class. Thanks for being willing to learn and work hard, and for generally having a good sense of humor about it all.

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] 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).

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