System Development: A Project Based Approach

Hossein Hakimzadeh, Robert Batzinger, Susan Gordon
Department of Computer and Information Sciences
Indiana University - South Bend
South Bend, IN 46615

hhakimza@iusb.edu

ABSTRACT

As computer science continues to move toward a more pragmatic market driven discipline, some of the traditional core topics in computing have become less popular. At-risk topics include compilers, file organizations, and operating systems. Viewed collectively, it appears that courses that either deal with the internal working of computers or courses that require system programming are being systematically removed from the undergraduate curriculum. This paper describes a project-based approach to seamlessly reintroduce system development topics in today's contemporary computer science curriculum.

1. INTRODUCTION

Most computer science programs base their curriculum on Computing Curricula recommendations put forth by the ACM /AIS / IEEE joint task force for computing curricula [1, 2]. With each new report, these recommendations move away from core computer science topics and toward contemporary topics such as cyber security, distributed computing, bioinformatics, and game programming.

Core topics such as compilers, file organizations, and operating systems appear to be at risk. There appears to be a growing trend where courses dealing with the internal working of computers or system software are being systematically replaced by contemporary electives. This trend can be seen in the 2001 and 2005 ACM/IEEE Computing Curricula recommendations.

This paper proposes increasing student exposure to system level topics by incorporating a series of project-based system development courses into the undergraduate curriculum. It will discuss a course in "*Advanced Database Systems*" as a case study for this approach. The objective of this course is to design and implement a database engine as the vehicle for fostering system development skills. The engine in turn serves as a platform for research and study of contemporary topics such as security, concurrency control, performance tuning, and data mining.

2. BACKGROUND

In an article published in 2002, Mary Shaw [3] advocates constructing the computer science curriculum around abstract themes that cut across the discipline. She further criticizes the teaching of "compiler construction" and "operating systems" as teaching "system-artifact dinosaurs". It is clear that members of the CC2001 Task Force were inspired by this article and consequently, the Computing Curricula recommendations clearly place less emphasis on system development topics such as compiler construction and file organizations.

Naturally, the CS curriculum at Indiana University South Bend has mirrored the CC2001 and CC2005 recommendations. The department is offering more courses such as graphics, networking, computer vision, game programming, software engineering, artificial intelligence, bioinformatics, and computer security. Although an excellent case can be made for teaching any of the above courses, the fact remains that students leave our program with significantly less exposure to system development skills. Two questions remain unanswered: First, **can computer science graduates thrive without these core skills?** And second, **how will the loss of these skills affect the next generation of system designers?**

Although, the concept of creating courses that cut across the discipline is appealing, we must be careful not to deprive the future computer scientists from an important body of knowledge fundamental to computing.

This paper profiles the implementation of a course in "*Advanced Database Systems*". The primary focus of this course is to study the inner working of system software, specifically how database management systems work. The course systematically leads the students through the design and implementation of a database engine called MiniDB [4] which is then used to explore advanced database topics.

Similar project-based courses in advanced database design were introduced by David DeWitt [5] and later extended by Mike Carey and Raghu Ramakrishnan [6] at the University of Wisconsin. Another system inspired by the DeWitt has been implemented by Albano *et al* [7].

In the remainder of this paper, we will summarize the structure of this course, and discuss some future directions.

3. THE MiniDB SYSTEM

The MiniDB system is developed as a semester long project to introduce the underlying theories, principles and practices needed to implement a simple but flexible database engine. The prerequisite for the course is an undergraduate database course that introduces the students to data modeling, the relational model, relational algebra, SQL, and additional topics such as transaction management, concurrency control, and data mining.

Conceptually, the course and its project (MiniDB) are divided into five phases: preparation; design and implementation of core algorithms; research in advance algorithms; implementation of advance algorithms; and presentation of the final project [4]. The interactions between these phases are shown in (Figure 1).

The above phases are planned such that in the first ten weeks of the semester, students construct a fully functional mini database engine. In the last five weeks, each student chooses to research one or more advanced topics in databases and integrates their research findings into their MiniDB implementation. The full poster will describe each phase of the course.

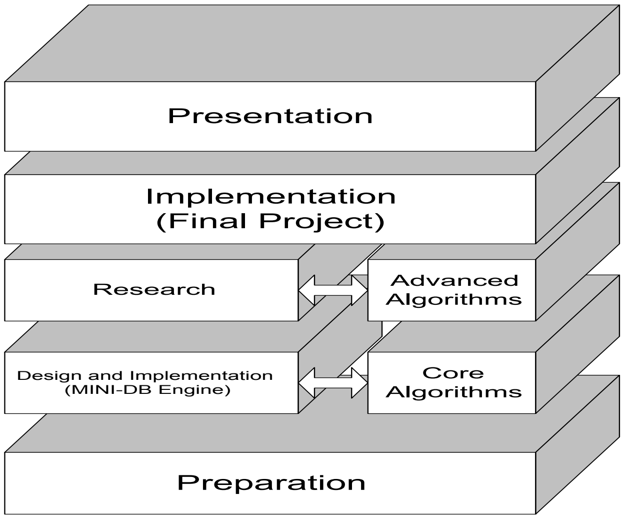
Presentation
Implementation (Final Project)
Research
Advanced Algorithms
Design and Implementation (MINI-DB Engine)
Core Algorithms
Preparation

Figure 1. MiniDB Conceptual Model

4. DESIGN CONSIDERATIONS

Initially, most students will find it difficult to envision creating a database engine from scratch. In order to guide the design and implementation process, the task is broken down into a series of deliverables which are due at two week intervals. Each deliverable serves two purposes. First, it seeks to incrementally construct new building blocks that move the project toward the goal of a working database engine. Second, it allows the students to systematically refine the previously constructed components. This ability to incrementally refine one's work is an essential yet underdeveloped skill among many students.

The remainder of this section discusses the project deliverables. These include the creation of access mechanisms, data definition language (DDL), data manipulation language (DML), relational algebra operators, meta-data, XML, and index structures.

4.1 Access Mechanisms

The goal of the first deliverable is to construct a series of classes for creating and manipulating simple data, index and meta-data files. These three classes provide the basis for creating a database table. The ***data file*** is a sequentially organized but randomly accessed file. The ***index file*** is a direct access file. The ***meta-data file*** is a sequential file that will maintain schema information about the database.

4.2 Data Definition and Manipulation

The goal of the second and third deliverables is to develop a data definition and data manipulation component for the MiniDB system. Relational algebra is chosen for this purpose. As shown in Figure 2, the basic relational algebra operations include select, project, cartesian product, join, union, intersection, and difference are implemented.

```
Class Mini_Rel_Algebra {
        bool create(relation_name, schema);
        bool insert(relation_name, attribute_list, value_list);
        bool delete(relation_name, attribute_name, condition,
                        attribute_value);
        bool modify(relation_name, search_attribute_name,
                        condition, search_attribute_value,
                        modify_attribute_list,
                        modify_value_list);
        result_rel select(relation_name, attribute_name,
                        condition, attribute value);
        result_rel project(relation_name, attribute_list);
        result_rel cartesian_product(relation_1, relation_2);
        result_rel join(relation_1, relation_2, condition_list);
        result_rel union(relation_1, relation_2);
        result_rel intersect(relation_1, relation_2);
        result_rel difference(relation_1, relation_2);
}
```

Figure 2. Relational Algebra Operations

4.3 Advanced Algorithms

The final phase of the course involves the creation and integration of advanced components on top of the basic MiniDB engine. During past offerings of this course, students have been able to develop algorithms for concurrency control, database security, access control, database integrity, external sorting, data mining, join optimization, distributed databases, and deductive databases [4].

CONCLUSION

This paper describes the design and implementation of a database engine as a vehicle for reinforcing system development topics in the computer science curriculum. Further, this paper describes the use of a project-based approach to systematically develop and refine code components for the database engine known as MiniDB. The project enables the students to successfully integrate the system development with the research and implementation of advanced database topics.

The above project-based, system development approach can be applied to other courses such as computer networks, computer graphics, and computer security. For example, a computer networking course can be augmented such that, through a series of assignments, students finish the course with their own MiniNetwork API based on the OSI model. Students in computer graphics could build their own MiniGameEngine. Finally, students in computer security could build a simple vulnerability scanner. As the computer science curriculum continues to refine and redefine itself, we hope that system development will regain more prominence in the curriculum.

REFERENCES

[1] IEEE/ACM-CS Computing Curricula 2001,Computer Science, Final Report, The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, December 15, 2001.
[2] ACM/AIS/IEEE-CS, Computing Curricula 2005, by The Joint Task Force for Computing Curricula 2005, 30 September 2005.
[3] Shaw, M., "We can teach software better." Computing Research News 4(4):2-12, September 1992.
[4] Hakimzadeh, H., "MINI-DB: project website" www.cs.iusb.edu/minidb
[5] DeWitt, D., the Minirel project. A database course project that involved building a small relational DBMS.
[6] Carey, M., Ramakrishnan, R., the Minibase project. Extension and redesign of Minirel project. www.cs.wisc.edu/coral/mini_doc/project.html
[7] Albano, A., JRS (Java Relational System) is a relational DBMS implemented in Java and designed for educational use. Accessed on the web on March 2008, www.di.unipi.it/~albano/JRS/project.html