DATABASE CURRICULUM ISSUES FOR FOUR-YEAR IT PROGRAMS

Joseph J. Ekstrom\textsuperscript{1} and Stephen R. Renshaw\textsuperscript{2}

Abstract – IT programs are facing the challenge of integrating coherent programs from topics that are parts of other domains. Database topics have been identified as one of the core areas of an IT curriculum. IT programs have typically adopted Database texts and approaches used in other disciplines. However, there is a current debate in Information Systems and Computer Science about the structure and content of Database courses. With the advent of the World Wide Web and browser-based computing, the use models for both the design and access of database systems have changed radically. Current best practice indicates the use of a multi-tiered deployment model for systems to facilitate scalability and manageability. Conceptual modeling with UML and a close binding of the system design to the real-world objects greatly reduces the need for traditional normal-form optimizations in relational database designs. The emphasis on use-case driven, iterative system development has changed the dynamics of database design in the system development process. An argument is made that the traditional database curriculum along with much of the traditional operating systems and programming language curriculum should be integrated into a system development sequence that presents the necessary concepts in the context of current system development processes for Information Technology students. We present our proposed curriculum as a work-in-progress to document the IT perspective and to elicit comment from the academy.

Index Terms – Database, Curriculum, Information Technology

INTRODUCTION

There is an emerging consensus among Information Technology programs that the core of an IT curriculum consists of Programming, Database, Web Technologies, Networking and possibly Human Computer Interfacing [1]. IT programs have typically implemented a traditional CS/IS Database course in the junior year [2] [3]. These courses have either been taught by IT faculty, or the students have been required to take appropriate courses from the CS or IS departments. At BYU we have taken the first approach. As we have gained experience teaching the course in the context of our IT curriculum we have observed several problems [4] that we reported at the CITC III conference in the spring of 2002. As we continue to work in the context of SITE to define and codify IT curriculum, we thought it wise to present our current thinking on database curriculum to a wider audience. Since there is no consensus IT database curriculum defined as yet, we examine the latest curriculum from CS and IS, then propose modifications that can form the basis for an IT database curriculum.

There is a current debate about the structure and content of Database courses [5] [6]. This discussion mirrors many of the issues and concerns we have encountered in implementation of the IT curriculum at BYU. We do not have the issues of database teaching inertia mentioned in [5] because BYU Information Technology is a new program [7] [8]. However, the issues of the relationship of database concepts to the other courses, especially web system development, have been encountered during our first year.[4] We have also discovered that the learning styles of our students are diverse and that we must be more flexible in our approach to teaching [9]. One of the major goals of our program is to integrate the theory into hands-on projects as early and as often as possible. We believe that a key differentiator between Information Technology and the other disciplines in computing is the emphasis on system integration, that is, the creation of operational systems from various components. As we discussed in [4], it is necessary to integrate the current tools for system development and management into the curriculum if we are to prepare our students to participate in the best practices currently in use. In this paper we present the issues of integrating system development best practices and database concepts with web systems development into curriculum. A major thrust of our effort has been the introduction of team projects into many of our courses from the sophomore level on, and an attempt to introduce the concepts of iterative development and object-oriented modeling early in the sophomore year so that the concepts and tools are available to the students during their entire junior core. This approach gives the students significant understanding of the issues of working in teams while producing a well defined result even before the capstone sequence in the senior year. However, the desire to introduce project experience early, along with the sequencing issues discussed in [4], make it impossible to cover database topics in the traditional topic-oriented approach of a database class, since a database component is required for elementary forms processing in web systems. Some of these issues are related to the nature of IT as a discipline, though others in the computing community have described similar challenges [5]

\textsuperscript{1}Joseph J. Ekstrom, Brigham Young University, School of Technology, 265 CTB, Provo, UT 84602 jekstrom@byu.edu
\textsuperscript{2}Stephen R. Renshaw, Brigham Young University, School of Technology, 265 CTB, Provo, UT 84602 srensh@et.byu.edu


ETD343-1
In this paper we first describe the problems that initially led us to consider significant changes in the curriculum. We then examine the rather limited literature on the subject of modern database curricula for clues to solutions. Next we derive a framework for describing and discussing the curriculum and pedagogy issues using the CC2001 [10] as a basis. Using this framework we classify our original curriculum for what CC2001 calls the intermediate level and look at the concept areas required to achieve a project orientation where best practices and tools can be introduced earlier in the curriculum, even in the second year, and relate them to the current state of the art as represented by the Rational Unified Process and its associated tools. Using the framework and concept areas, we describe the latest revision of the database and web systems curriculum at BYU and how it addresses the shortcomings of our earlier pedagogical approach. In conclusion we discuss the transitional issues we are facing, our assessment approach and concerns about the implementation of the planned changes.

THE PROBLEM

As recent arrivals in academia from industry, the authors have had experience with some of the methodologies and tools that define the state of the art in system and software development. In [4] we described some of the issues we discovered in our first foray into the teaching of the curriculum defined for the new BYU IT program. The following expands and clarifies those issues:

1. The background of students was extremely varied.
   a. We had professional Perl programmers in the same class with students who had had a single Java programming class.
   b. We had system administrators and students who didn’t know what a process was.
2. Web services use databases and both of them require the services of operating systems and networks. It is nearly impossible to discuss any of these four as intermediate topics without discussing the fundamentals of the others.
3. The textbooks have a definite Software creation orientation, where an IT program needs to emphasize system integration. All computing programs, whether Computer Science, Information Systems, Computer Engineering or Information Technology are based on the same fundamentals; however, the application and focus is different. We observed this in our analysis of the Computer Networks class last year,[12] and we see the same situation in the rest of the curriculum. Some of the textbooks work for multiple disciplines, but the labs required are different.
4. The students needed to be introduced to requirements analysis and modeling so that they could communicate the details of their projects to the team members and the rest of the class.
5. There was no notion of best practices and associated tools in the topic-oriented curriculum.
6. Iterative development and system evolution concepts were absent from the curriculum.

RECENT DATABASE CURRICULUM LITERATURE

The current literature on database curriculum is very sparse. As was observed in [5], after a 10 year lag, we have entered a time when it is not clear that the standard curriculum in databases is adequate to the current state of affairs. SIGCSE 2000 had a few papers on database curriculum [5] [6] [11] [12] . References [13] and [14] are more recent and discuss the introduction of database topics into the context of a web systems development class. Ms. Robert indicates that there are some papers in preparation for SIGCSE 2003 [15]. However, all of these papers either provide case studies of merging concepts from the database class into a web systems development class, or simply describe the problem and provide evidence that the problem is real. None really addresses the problem of sequencing a database curriculum across a wider curriculum. However, [14] did give us encouragement that our thinking was not out of line with the feelings of at least some other educators.

CC2001 BODY OF KNOWLEDGE AND PEDAGOGY

The IEEE/ACM joint task force on Computer Science Curriculum 2001 takes the approach of breaking the discipline into a set of concepts that cover the “body of knowledge” that is computer science .[10]. Using this approach allows one to analyze a curriculum for coverage independent of the sequence that the concepts are presented in. We do not mean to imply that the Body of Knowledge for Information Technology or pedagogical approaches will be identical to that of computer science, but there is clearly a great deal of overlap. This is particularly true in the areas of IT that are directly involved with software. In addition, it is clear that any IT system deployment is likely to have a significant software component.
We believe that it is useful to use the CC2001 as a baseline for evaluating the coverage of our IT offering. In particular, the 8 areas highlighted in Table I are of interest. We observed in [12] that the lecture component of a CS networking class and that of an IT networking class were nearly identical. In fact we have chosen to use the same text [16]. However the labs were very different. While CS labs focused on programming tasks for network stacks and had one lab on device configuration, our IT curriculum has one programming oriented lab and many labs of installation, configuration, scripting components to achieve a result and troubleshooting. We have had similar experiences in comparing our web systems development class to a class taught in the Information Systems program in the Business School. The practicum for IT includes significantly more installation, configuration, and system testing components than those of the other programs covering similar topics. This similarity of topics does, however, allow us to use the conceptual framework for both the body of knowledge and pedagogy with little change from that defined in CC2001 in spite of the differences in practicum. In practice, the coverage of the same topics will be oriented toward system integration, test and deployment rather than component development. An IT curriculum could start with the CC2001 structure and contents, include a significant subset of what is already there, then add and delete knowledge areas. An example would be that an IT program would not have a Software Engineering area but rather a Systems Engineering area. The concept areas would include methods for making and managing component and deployment decisions as well as many concept areas in the CC2001 SE area.

**Table I**

<table>
<thead>
<tr>
<th>Computer Science Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Structures (DS)</td>
</tr>
<tr>
<td>Programming Fundamentals (PF)</td>
</tr>
<tr>
<td>Algorithms and Complexity (AL)</td>
</tr>
<tr>
<td>Architecture and Organization (AR)</td>
</tr>
<tr>
<td>Operating Systems (OS)</td>
</tr>
<tr>
<td>Net-Centric Computing (NC)</td>
</tr>
<tr>
<td>Programming Languages (PL)</td>
</tr>
<tr>
<td>Human-Computer Interaction (HC)</td>
</tr>
<tr>
<td>Graphics and Visual Computing (GV)</td>
</tr>
<tr>
<td>Intelligent Systems (IS)</td>
</tr>
<tr>
<td>Information Management (IM)</td>
</tr>
<tr>
<td>Social and Professional Issues (SP)</td>
</tr>
<tr>
<td>Software Engineering (SE)</td>
</tr>
<tr>
<td>Computational Science and Numerical Methods (CN)</td>
</tr>
</tbody>
</table>

Table II is a map of the pedagogical approaches enumerated in CC2001. This conceptual framework has been a valuable tool to solidify our thinking and as a jumping off point for discussion. The concept areas and approach map are self explanatory for the purposes of this paper. The reader is referred to CC2001 for additional details. These specific artifacts were selected (in **bold**) because they represent the hierarchy above the Information Management area, which includes the database concept areas. Table II presents the approaches that the Computer Science community views as viable. CC2001 is very careful to explain that each of these approaches has advantages and disadvantages. There is no consensus on the “best” approach.

### Table II

**CC2001 Pedagogical Approach Map**

<table>
<thead>
<tr>
<th>Introductory Courses</th>
<th>Imperative first</th>
<th>Objects first</th>
<th>Functional first</th>
<th>Breadth first</th>
<th>Algorithms first</th>
<th>Hardware first</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Courses</td>
<td>Topic-based approach</td>
<td>Compressed approach</td>
<td>Systems-based approach</td>
<td>Web-based approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Courses</td>
<td></td>
<td>Additional courses used to complete the undergraduate program</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
by the major accreditation committees is extremely helpful. Knowing topics and an estimate of the minimum number of lecture hours required makes a significant difference in the effort required to define a course, even if the topics are modified to meet specific needs. We have included the complete listing of CC2001 concept areas as Appendix A.

| Table III  
| IM CONCEPT AREAS  

**IM. Information Management (10 core hours)**
- IM1. Information models and systems (3)
- IM2. Database systems (3)
- IM3. Data modeling (4)
- IM4. Relational databases
- IM5. Database query languages
- IM6. Relational database design
- IM7. Transaction processing
- IM8. Distributed databases
- IM9. Physical database design
- IM10. Data mining
- IM11. Information storage and retrieval
- IM12. Hypertext and hypermedia
- IM13. Multimedia information and systems
- IM14. Digital libraries

| Table IV  
| EXPANSION OF DATA MODELING AREA  

**IM3. Data modeling [core]**

Minimum core coverage time: 4 hours

*Topics:*
- Data modeling
- Conceptual models (including entity-relationship and UML)
- Object-oriented model
- Relational data model

*Learning objectives:*
1. Categorize data models based on the types of concepts that they provide to describe the database structure—that is, conceptual data model, physical data model, and representational data model.
2. Describe the modeling concepts and notation of the entity-relationship model and UML, including their use in data modeling.
3. Describe the main concepts of the OO model such as object identity, type constructors, encapsulation, inheritance, polymorphism, and versioning.
4. Define the fundamental terminology used in the relational data model.
5. Describe the basic principles of the relational data model.
6. Illustrate the modeling concepts and notation of the relational data model.

**BEST PRACTICE IN IT SYSTEMS DEVELOPMENT**

An example of the current state of the art in System Development is the Rational Unified Process (RUP)[17]. This methodology has evolved from the work of several groups from all over the world. The key best practices according to RUP are:

1. Develop software iteratively.
2. Manage Requirements.
3. Use component-based architectures.
4. Visually model software.
5. Verify software quality.
6. Control changes to software.

The methodology focuses on software development rather than systems development, but from an IT perspective many of the techniques apply to systems development of any kind. The Unified Modeling Language (UML) [18] allows one to
describe hardware as well as software components. One must build and deploy most computer systems iteratively, small deployment with early software, revise, deploy, revise, deploy... Those of us who have built and deployed industrial systems know how much they must evolve. The RUP is a process designed to manage the reality of evolving systems. We know the problems.; we have ways of dealing with them, yet this process orientation does not fit well into a purely topic-oriented approach. We must prepare our students with UML modeling experience in all of the systems related classes, especially web and database systems. The tools for using these techniques have been maturing rapidly. Both Rational Systems™ and TogetherSoft™ have complete suites for diagramming the systems and then building and managing software development. As a VP of engineering, the principal author was willing to spend several thousand dollars per engineer so that they would have a common language and tools for describing systems. We will do a great service to both our students and their future employers if we prepare them with the latest tools and techniques for system development and evolution.

**Original Curriculum**

Our original curriculum is described in [4]. For the purposes of this paper it is sufficient to say that our introductory courses follow what CC2001 would describe as a Hardware first approach with significantly more electronics and hands-on with analysis tools than a typical CS program. Our intermediate approach was purely topic-based and follows a similar concept presentation sequence as the topic-based curriculum for OS, IM, NC, and AR with a smattering of PL, HC, SP, and SE. Indeed, our IT program covered the CC2001 core in OS, IM, PF, NC, AR, HC. (refer to Table I above for the meaning of these acronyms). In CC2001 the database topics are included in IM, information management.

**Proposed Curriculum**

After we began to implement the changes described in [4], we realized that we had not adequately addressed a significant problem. Indeed the problem of students with insufficient background to understand the basic functioning of a web server and its relationship to the browser was the first major flag that appeared when the web systems class was taught for the first time. As we discussed this problem as a faculty, we also realized that the class dealing with human computer interfacing would benefit from students having experience with web systems as a prerequisite. The HCI class was taught in the sophomore year and web systems was taught in the junior year. We had already decided that we would teach introductory concepts of operating systems (OS), database (IM), and networking (NC) in the web systems class, and that it would be a prerequisite for the database class. With this in mind we decided to create a web systems class at the sophomore level and move the HCI class to the junior core. This also has the advantage that by using this class as gateway to the junior core, instructors can assume a certain level of proficiency in the fundamentals of IT platform technologies.

To summarize, we modified the original curriculum by:

1. Creating a *Fundamentals of web-based Information Technology* class that incorporates the introductory concepts from OS, NC, IM, SP, and SE, including elementary data modeling and use cases.
2. Removing the concepts taught in 1 from the corresponding classes in the junior core, leaving time for better intermediate coverage of the material.
3. Moving HCI from the sophomore to the junior year and the fundamentals course to the sophomore year.
4. Adding a web emphasis to the HCI course.
5. Emphasizing modeling as a tool for management of complexity throughout the system development process. A strong introduction is given within the sophomore class and usage is encouraged within the junior and senior level classes.
6. Including team projects in each of the courses, including the sophomore level course.
7. Including the use of state of the art tools through educational copies and the use of the 30 day free trial of the software. (Vendors who allow free academic use of their products get a big advertising bonus here.).
8. Providing deeper coverage of enterprise systems deployment in the database class.

The resulting curriculum is included as Appendix B.

**Issues and Implementation of the Proposed Curriculum**

We have several concerns in implementing this new curriculum such as the transition of students from the old curriculum to the new, the need to integrate the teaching of experienced and novice students, emphasis on life-long learning and assessment of the effectiveness of the curriculum so it can be improved.
In order to minimize impact on the overall curriculum, we will leave the networking and operating systems classes unchanged initially. Students who have had the new IT fundamentals--database sequence will be significantly better prepared for these two classes and as time progresses the courses could be modified to focus on more advanced topics. It is a challenge to transition between programs and be fair to all of the students. This problem is compounded at BYU in that most of our students serve a 2-year mission for the sponsoring institution sometime between their Freshman and Junior years. This stretches any transition to at least 3 years.

This new curriculum will be implemented as changes to the IT fundamentals (web systems development) course, the database principles course, and the computer networks course on a trial basis by the primary author in the coming year. We can introduce the proposed approach without impacting other courses as long as the required material is covered. It is our intent to evolve toward a seminar based approach oriented around the team projects during the next 2-4 years. Team projects are normally proposed and implemented by the students into full working prototypes. We have experimented with having students teach lectures on topics for which they have particular expertise in the past. The web-based classes make this a requirement since many of the students have significantly more experience with some of the specific technologies than any of the faculty. We believe that the best way to keep abreast of the changes in the technologies involved is to leverage every asset available to the class, especially the experience of its members.

However, there is often a tendency for beginning students to resist when they are given a less structured course. Some become frustrated that there are many possible answers, some of them better than others, but none of them “correct” in the sense of a traditional textbook question. We will monitor student attitudes and progress so we can make adjustments to the curriculum ensure We will continue to emphasize to the students that much of the lack of detailed structure in the course is actually part of the learning. This problem will be even more pronounced for students returning after a two-year absence. The student teams’ choices of projects will strongly influence the details of how topics are covered from year to year.

We realize that assessment of progress using these less structured approaches is more difficult. We must be concerned from the beginning that we measure learning effectiveness during the process of introducing the new curriculum. We are especially concerned about how we measure the effectiveness of the course as it evolves on top of evolving technology. We cannot compare the resulting web sites because the tools can make so much difference in the appearance of the site that it is difficult to measure actual learning rather than the effectiveness of a new tool. Team dynamics are also a concern for assessment. We must find ways to assess if a student is a “team player” as defined within current industry norms.

**Assessment**

To assess if the new curriculum is effective in realizing the results expected, several aspects must be considered:

1. Will the student learning outcomes at least stay the same or become better? Student outcomes are the main objective of the curriculum and must be maintained through any changes. Data from previous courses taught in the form of finals and projects will be compared with finals and projects within the new curriculum to determine if the outcomes are still being met.
2. Will the curriculum be more effective in achieving learning outcomes in that it is easier to deliver? This measure is mostly in the opinion of the instructors that are presenting the courses. Since they are being taught by the same instructors as the previous courses the feeling of the course by the instructors will be easy to gather.
3. Will class evaluations show better student attitudes towards courses and instructors with the new curriculum? Student evaluations are taken near the end of every course. The results of these evaluations for the new curriculum will be compared with the previous results for the courses affected by the changed curriculum. Normally if students rate a course highly they feel as though they have learned useful material without extreme effort. The hope is that this will show that the new curriculum will deliver the same material in a manner that facilitates a better learning environment.
4. Will team participation be facilitated as well or better in the new curriculum? Team participation in projects is essential within the IT field and is used throughout the BYU IT program. Many of the courses affected by the changed curriculum have used peer grading within the team to help the instructors know the participation level of each student. Since with the new curriculum students should be better prepared for the Junior level classes it should help raise the level of participation within team projects. The peer scoring from previous courses will be compared with the new courses to determine the level of team participation.
CONCLUSION

We believe that it is possible to significantly improve student learning through refactoring the curriculum so that more fundamental IT concepts are introduced in the sophomore year. The students get experience building a website as a team early in the curriculum. We introduce database concepts sufficient to do SQL queries on a single table and sufficient operating systems and networking concepts for the students to install and manage a simple web server. The students build a website and host it as a team project. This course gives an early working knowledge of integrating aspects of Database, web technology, OS, and networking. It will serve as a major component of the baseline for student proficiency before admittance into the IT professional program. The increased uniformity in the preparation of students entering the junior core, in addition to the coverage of 2-6 lectures from the intermediate classes, allows deeper coverage in those classes. The impact will be especially beneficial in the intermediate database curriculum since students will enter the course familiar with UML and modeling as well as be familiar with the mechanics of SQL and web-based interfaces to database systems. We will be able to give significantly deeper coverage of enterprise and internet database deployment models. It is our intention to formally assess the effects of these changes and report our results. We hope that the presentation of these early results will assist other IT programs as they deal with these issues and that others will share their thinking on the subjects presented.

REFERENCES

[7] Lunt, Barry, et. al., Designing an IT Curriculum: The Results of the First CITC Conference, ASEE 2002 Session 2650
[14] Treu, Kevin, To Teach the Unteachable Class: An Experimental Course in Web-Based Application Design, SIGCSE’02, Covington, Kentucky, USA,
## APPENDIX A: COMPUTER SCIENCE BODY OF KNOWLEDGE

### DS. Discrete Structures (43 core hours)
- **DS1. Functions, relations, and sets (6)**
- **DS2. Basic logic (10)**
- **DS3. Proof techniques (12)**
- **DS4. Basics of counting (5)**
- **DS5. Graphs and trees (4)**
- **DS6. Discrete probability (6)**

### PF. Programming Fundamentals (38 core hours)
- **PF1. Fundamental programming constructs (9)**
- **PF2. Algorithms and problem-solving (6)**
- **PF3. Fundamental data structures (14)**
- **PF4. Recursion (5)**
- **PF5. Event-driven programming (4)**

### AL. Algorithms and Complexity (31 core hours)
- **AL1. Basic algorithmic analysis (4)**
- **AL2. Algorithmic strategies (6)**
- **AL3. Fundamental computing algorithms (12)**
- **AL4. Distributed algorithms (3)**
- **AL5. Basic computability (6)**
- **AL6. The complexity classes P and NP**
- **AL7. Automata theory**
- **AL8. Advanced algorithmic analysis**
- **AL9. Cryptographic algorithms**
- **AL10. Geometric algorithms**
- **AL11. Parallel algorithms**

### AR. Architecture and Organization (36 core hours)
- **AR1. Digital logic and digital systems (6)**
- **AR2. Machine level representation of data (3)**
- **AR3. Assembly level machine organization (9)**
- **AR4. Memory system organization and architecture (5)**
- **AR5. Interfacing and communication (3)**
- **AR6. Functional organization (7)**
- **AR7. Multiprocessing and alternative architectures (3)**
- **AR8. Performance enhancements**
- **AR9. Architecture for networks and distributed systems**

### OS. Operating Systems (18 core hours)
- **OS1. Overview of operating systems (2)**
- **OS2. Operating system principles (2)**
- **OS3. Concurrency (6)**
- **OS4. Scheduling and dispatch (3)**
- **OS5. Memory management (5)**
- **OS6. Device management**
- **OS7. Security and protection**
- **OS8. File systems**
- **OS9. Real-time and embedded systems**
- **OS10. Fault tolerance**
- **OS11. System performance evaluation**
- **OS12. Scripting**

### NC. Net-Centric Computing (15 core hours)
- **NC1. Introduction to net-centric computing (2)**
- **NC2. Communication and networking (7)**
- **NC3. Network security (3)**
- **NC4. The web as an example of client-server computing (3)**
- **NC5. Building web applications**
- **NC6. Network management**
- **NC7. Compression and decompression**
- **NC8. Multimedia data technologies**
- **NC9. Wireless and mobile computing**

### PL. Programming Languages (21 core hours)
- **PL1. Overview of programming languages (2)**
- **PL2. Virtual machines (1)**
- **PL3. Introduction to language translation (2)**
- **PL4. Declarations and types (3)**
- **PL5. Abstraction mechanisms (3)**
- **PL6. Object-oriented programming (10)**
- **PL7. Functional programming**
- **PL8. Language translation systems**
- **PL9. Type systems**
- **PL10. Programming language semantics**
- **PL11. Programming language design**

*Note: The numbers in parentheses represent the minimum number of hours required to cover this material in a lecture format. It is always appropriate to include more.*

### HC. Human-Computer Interaction (8 core hours)
- **HC1. Foundations of human-computer interaction (6)**
- **HC2. Building a simple graphical user interface (2)**
- **HC3. Human-centered software evaluation**
- **HC4. Human-centered software development**
- **HC5. Graphical user-interface design**
- **HC6. Graphical user-interface programming**
- **HC7. HCI aspects of multimedia systems**
- **HC8. HCI aspects of collaboration and communication**

### GV. Graphics and Visual Computing (3 core hours)
- **GV1. Fundamental techniques in graphics (2)**
- **GV2. Graphic systems (1)**
- **GV3. Graphic communication**
- **GV4. Geometric modeling**
- **GV5. Basic rendering**
- **GV6. Advanced rendering**
- **GV7. Advanced techniques**
- **GV8. Computer animation**
- **GV9. Visualization**
- **GV10. Virtual reality**
- **GV11. Computer vision**

### IS. Intelligent Systems (10 core hours)
- **IS1. Fundamental issues in intelligent systems (1)**
- **IS2. Search and constraint satisfaction (5)**
- **IS3. Knowledge representation and reasoning (4)**
- **IS4. Advanced search**
- **IS5. Advanced knowledge representation and reasoning**
- **IS6. Agents**
- **IS7. Natural language processing**
- **IS9. AI planning systems**
- **IS10. Robotics**

### IM. Information Management (10 core hours)
- **IM1. Information models and systems (3)**
- **IM2. Database systems (3)**
- **IM3. Data modeling (4)**
- **IM4. Relational databases**
- **IM5. Database query languages**
- **IM6. Relational database design**
- **IM7. Transaction processing**
- **IM8. Distributed databases**
- **IM9. Physical database design**
- **IM10. Data mining**
- **IM11. Information storage and retrieval**
- **IM12. Hypertext and hypermedia**
- **IM13. Multimedia information and systems**
- **IM14. Digital libraries**

### SP. Social and Professional Issues (16 core hours)
- **SP1. History of computing (1)**
- **SP2. Social context of computing (3)**
- **SP3. Methods and tools of analysis (2)**
- **SP4. Professional and ethical responsibilities (3)**
- **SP5. Risks and liabilities of computer-based systems (2)**
- **SP6. Intellectual property (3)**
- **SP7. Privacy and civil liberties (2)**
- **SP8. Computer crime**
- **SP9. Economic issues in computing**

### SE. Software Engineering (31 core hours)
- **SE1. Software design (8)**
- **SE2. Using APIs (5)**
- **SE3. Software tools and environments (3)**
- **SE4. Software processes (2)**
- **SE5. Software requirements and specifications (4)**
- **SE6. Software validation (3)**
- **SE7. Software evolution (3)**
- **SE8. Software project management (3)**
- **SE9. Component-based computing**
- **SE10. Formal methods**
- **SE11. Software reliability**
- **SE12. Specialized systems development**

### CN. Computational Science (no core hours)
- **CN1. Numerical analysis**
- **CN2. Operations research**
- **CN3. Modeling and simulation**
- **CN4. High-performance computing**

---

**2003 CIEC Conference**  
**ETD343-8**  
**January 28 thru 31, 2003, Tucson, Arizona**
### APPENDIX B: BYU DATABASE CURRICULUM

#### Web Systems Development: IT 210

<table>
<thead>
<tr>
<th>Topics:</th>
<th>Labs:</th>
</tr>
</thead>
</table>
| 1. Fundamentals:  
  a. What’s a model? (IM3)  
  b. What’s a computer? (AR6)  
  c. What’s an operating system? (OS1-5)  
  d. What’s a Database? (IM1-2)  
  e. What’s a network? (NC1-2)  
  f. What’s a system? (NC4)  
| 1. Intro to Modeling tools  
  a. UML, Java IDE  
  b. Projects, Tasks, Resources  
| 2. Human Factors (HC1-4)  
| 3. System Life Cycle (SP2-4, SE5-8, IM3)  
  1. Business Context  
  2. Users & Customers  
  3. System Modeling and the UML  
  4. Use Cases  
  5. Requirements  
  6. Design  
  7. Development  
  8. Deployment  
  9. Maintenance  
| 2. How the web works (Client)  
  a. HTTP protocol, bits and bytes.  
  b. Static HTML  
  c. JavaScript Programming  
  d. Java Applets  
| 3. How the web works (Server)  
  a. Perl Programming (CGI)  
  b. Java Servelets  
| 4. How the web works (Dynamic Content)  
  a. PHP  
  b. ASP  
  c. JSP  
| 5. How the web works (Enterprise)  
  a. Middleware  
  b. J2EE  
  c. .NET  
| 6. Class Project  
  a. 10+ pages  
  b. Dynamic content  
  c. Single table database  
| 4. Web Technology (NC4, AR9)  
  a. HTML  
  b. Forms and CGI  
  c. Javascript  
  d. Perl Programming  
  e. PHP Programming  
  f. Java Server Pages  
  g. Middleware Systems  
  h. Database Integration  
| 1. Introduction to Microsoft Access  
  2. Relational Modeling  
  3. Implementing Model in Access  
| 5. Project (SE1-4, SP2-4)  
  a. Teams  
  b. Project Management  
  c. Case Studies  
  d. Do it for Class Project  
| 4. Web Technology (Dynamic Content)  
  a. PHP  
  b. ASP  
  c. JSP  
| 6. Class Project  
  a. 10+ pages  
  b. Dynamic content  
  c. Single table database  

#### Database Systems: IT 350

<table>
<thead>
<tr>
<th>Topics:</th>
<th>Labs:</th>
</tr>
</thead>
</table>
| 1. Data Modeling with UML (IM2)  
  2. Relational Data Model (IM4)  
  3. Database Schema (IM6)  
  4. Schema Optimization (IM6)  
  5. Data Manipulations/ Relational Algebra (IM5)  
  6. SQL (IM5)  
  7. Databases and Internet Integration (NC4, IM2,8)  
  8. Efficient Indexing (IM9)  
  9. Query Optimization (IM5)  
  10. Transaction Processing (IM7)  
  11. Object oriented databases  
  12. Security in database servers  
| 1. Introduction to Microsoft Access  
  2. Relational Modeling  
  3. Implementing Model in Access  
| 4. Web integration to Access DB using SQL  
  5. ODBC/JDBC and database systems  
  6. JDBC Programming  
  7. Final Team Project counts as 2-4 Labs  

*Note: CC2001 partial area coverage indicated after topics: (IM?, OS?, SE?...)*