computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

International Scientific Journal of Computing

ISSN 1727-6209

COMPUTING CURRICULA 2001 AND IDAACS

Allen B. Tucker

Bowdoin College, Brunswick, ME 04011, USA
allen@bowdoin.edu, www.bowdoin.edu/~allen

Abstract: *This paper discusses the relationship between recent advances in curricula in computer science in the US and intelligent data acquisition/advanced computer systems (IDAACS) research and development. The recent Computing Curricula 2001 Draft Report is used as a primary source of information about current developments in curriculum standards in the US.*

Keywords: *Computer science education, undergraduate curriculum, computer engineering, software engineering, intelligent data acquisition, advanced computer systems.*

1. INTRODUCTION

Some subject areas in undergraduate computer science curricula provide strong preparation for undergraduates who intend to conduct graduate study or professional work in the various areas of intelligent data acquisition and advanced computer systems. This paper examines the connection between the curricula and IDAACS.

Computing Curricula 2001 [1] is the most recent in a series of models that have guided American colleges and universities' undergraduate computer science and engineering degree programs for the last three decades. Earlier models include Curricula 91 [2], Curriculum 78 [3], and Curriculum 68 [4]. Curricula 2001 is distinguished from earlier models in five important ways:

1. It reflects the enormous technological changes of the last 10 years.
2. It reflects the major cultural, economic and pedagogical changes of the last 10 years.
3. It is international in scope.
4. It reflects the emergence of new subject areas.
5. It proposes separate curriculum models for the fields of computer science, computer engineering, software engineering, and information systems.

Curricula 2001 provides a model for designing undergraduate computer science programs by defining the following fourteen major subject areas as a basis:

Algorithms and Complexity
* Architecture and Organization
* Computational Science / Numerical Methods
Discrete Structures
Graphics and Visual Computing
* Human-Computer Interaction
* Information Management
* Intelligent Systems
Net-Centric Computing
* Operating Systems
Programming Fundamentals
Programming Languages
Software Engineering
Social and Professional Issues

This paper concentrates on those subject areas marked with an asterisk (*), since these are the ones most relevant to the interests of undergraduates who will pursue graduate study and careers in intelligent data acquisition and advanced computer systems.

By "Intelligent Data Acquisition/Advanced Computer Systems" (or IDAACS), we mean that field of research and development that designs and implements hardware and software systems that acquire data from sensors and other sources, interpret that data, and use that interpretation to help control a complex computer or network of computers. Such systems are "intelligent" because they interpret the raw sensory data before passing it on as a control. Such systems are "advanced" because they typically require complex computational interfaces for gathering information from a collection of heterogeneous devices. They are also advanced because the computers they control are highly parallel and complex in their hardware and software designs.

Examples of IDAACS are abundant. For instance, a bank's ATM system acquires data from users at a collection of ATM machines, processes that data in real time by retrieving relevant information from a central database of bank accounts, and responds to each user in accordance with his/her request. For another example, an embedded computer in a commercial airplane is an IDAACS, gathering atmospheric and navigational data from a collection of sensors and ground controllers. It then interprets that data and uses it to help guide

the plane or advise the pilot making decisions during the flight.

2. COMPUTING CURRICULA 2001: OVERVIEW

The general direction taken by the ACM/IEEE Curricula 2001 model [CC2001] reflects the need to cover a broad range of subjects and programs. This section summarizes the computer science body of knowledge and its core topics that are required for all undergraduate programs, whether they are IDAACS-related or not. The metric used here is the "core hour," which is a single lecture hour for a subject within a typical 1-semester course that would have 40 lecture hours overall. To cover all 14 subject areas listed above, the model proposes 280 = 7 semester courses worth of required core hours. Here is how that subject matter is distributed (core hours are given in parentheses alongside each topic) across these areas.

• Algorithms and Complexity (31 core hours): This subject area includes some of the basic underlying theory for computer science, and includes algorithmic analysis (4), algorithmic strategies (6), fundamental computing algorithms (12), distributed algorithms (3), and basic computability theory (6) as required topics. Other topics in this subject area that are not in the core body of knowledge (but are usually covered in typical courses) include the complexity classes P and NP, automata theory, advanced algorithmic analysis, cryptographic algorithms, geometric algorithms, and parallel algorithms.

• * Architecture and Organization (36 core hours): This subject area mainly addresses the design and engineering part of the discipline, and includes digital logic and digital systems (6), machine level representation of data (3), assembly level machine organization (9), memory system organization and architecture (5), interfacing and communication (3), functional organization (7), multiprocessing and alternative architectures (3). Other topics in this subject area that are not in the core body of knowledge include performance enhancement and architecture for networks and distributed systems.

• * Computational Science and Numerical Methods (0 core hours): This subject area is an emerging part of the discipline and emphasizes the interconnections between computer science and the traditional science and engineering disciplines. It includes topics in numerical analysis, operations research, modeling and simulation, and high-performance computing.

• Discrete Structures (43 core hours): This subject area was listed among the mathematics requirements in earlier curriculum models. However, since many computer science departments now teach this subject themselves, and since there is an increasing awareness of the need for integration of these topics within the traditional computer science topics, it is now part of the core body of knowledge in computer science. Topics include functions, relations, and sets (6), logic (10), proof techniques (12), counting and combinatorics (5), graphs and trees (4), and discrete probability (6).

• Graphics and Visual Computing (5 core hours): This area is the home for topics in computer graphics and visualization technologies. Required core topics include fundamental techniques in graphics (2), graphic systems (1), and graphic communication (2). Other topics in this area that are not in the core include geometric modeling, basic and advanced rendering, animation, visualization, virtual reality, and computer vision.

• * Human-Computer Interaction (6 core hours): This area includes coverage of the foundations of human-computer interaction (6), human-centered software evaluation and design, graphical user-interface design and programming, HCI aspects of multimedia systems, and HCI aspects of collaboration and communication.

• * Information Management (10 core hours): This area covers topics in management information systems and models (3), database systems (3), and data modeling (4). Other topics in this area not included in the core include relational databases, database query languages, relational database design, transaction processing, distributed databases, physical database design, data mining, information storage and retrieval, hypertext and hypermedia, multimedia information and systems and digital libraries. This area is a renaming and updating of the area traditionally called "database and information retrieval," adding new topics that have emerged out of the World Wide Web.

• * Intelligent Systems (10 core hours): This area is a renaming and updating of the traditional area called "artificial intelligence," and covers fundamental issues in intelligent systems (1), search and constraint satisfaction (5), and knowledge representation and reasoning (4). Other topics in this area include advanced search, knowledge representation and reasoning, agents, natural language processing, machine learning and neural networks, AI planning systems, and robotics.

• Net-Centric Computing (15 core hours): This

new area was added as a direct result of the emergence of the World Wide Web in the 1990s, and includes topics like an introduction to net-centric computing (2), communication and networking (7), network security (3), and the web as an example of client-server computing (3). Additional topics in this area include building web applications, network management, data compression and decompression, multimedia data technologies, and wireless and mobile computing.

• * Operating Systems (18 core hours): This area updates the traditional subject of operating systems, and the core material includes an 0verview of operating systems (2), operating system principles (2), concurrency (6), scheduling and dispatch (3), and memory management (5). Additional topics in this area that would occur in a course on operating systems include device management, security and protection, file systems, real-time and embedded systems, fault tolerance, performance evaluation, and scripting.

• Programming Fundamentals (54 core hours): This new area was viewed as an implicit prerequisite topic in prior curriculum models, and now it is explicitly defined via the following core lecture topics: fundamental programming constructs (9), algorithms and problem-solving (6), object-oriented programming (10), fundamental data structures (14), recursion (6), event-driven and concurrent programming (4), and using APIs (5).

• Programming Languages (6 core hours): This area includes an overview of programming languages (2), fundamental issues in language design (1), virtual machines (1), and an introduction to language translation (2). Additional topics in this area include a detailed study of language translation systems, type systems, models of execution control, declaration, modularity, and storage management, programming language semantics, programming paradigms, and language-based constructs for parallelism.

• Social and Professional Issues (16 core hours): This new area was treated not as a subject area in earlier curriculum models, but as a pervasive theme that crossed all subject areas. It includes as core topics a history of computing (1), the social context of computing (3), methods and tools of analysis (2), professional and ethical responsibilities (3), risks and liabilities of computer-based systems (2), intellectual property issues (3), and privacy and civil liberties (2). Additional topics in this area include computer crime, economics of computing, and philosophical issues.

• Software Engineering (30 core hours): This area includes the core topics of software processes (2), requirements and specifications (6), design (6), validation (6), software evolution (4), project management (4), and software tools and environments (2). Additional topics that round out this area include component-based computing, formal methods, software reliability, and specialized systems development.

Overall, a wider range of subjects is covered by the Curricula 2001 model than its predecessors, due to the rapid emergence of new computing technologies and research areas. At the same time, each topic is necessarily covered in far less depth than in prior curricula. Moreover, this model contains a significant shift in emphasis away from the theory and principles, and toward the systems and applications of the discipline. While undergraduate programs in computer science will be greatly challenged to meet the demands of the 21st century, this new model will provide important guidance for them to evolve in a contemporary fashion.

3. IDAACS-RELATED SUBJECT MATTER

The relationship between the Curricula 2001 proposal and the curricular needs of IDAACS can be analyzed in at least two different ways. One basis for analysis is to evaluate the total number of core lecture hours in the core body of knowledge that are relevant to IDAACS. Another basis is to identify selected courses that have been proposed in the Curricula 2001 draft report that would be considered essential core material for students who plan to pursue IDAACS-related graduate study.

In the first case, we assume that the subject areas marked with an asterisk in the above description are most closely related to IDAACS. The number of core lecture hours in these six areas that are required for an undergraduate curriculum is summarized as follows:

Architecture and Organization	36
Computational Science / Numerical Methods	0
Human-Computer Interaction	6
Information Management	10
Intelligent Systems	10
Operating Systems	18

This amounts to 80 core hours out of a total of 280 (28%), which is the equivalent of two semester courses, that are directly related to IDAACS subjects.

In the second case, we examine four sample courses that are proposed in the Curricula 2001

draft report and seem to be particularly well-suited for students who intend to pursue IDAACS-related graduate study or professional careers. Each one of these courses is a 40 lecture hour course, and contains some material from the core body of knowledge described above. The four courses are titled:

Computer Architecture
Databases
Information Management
Intelligent Systems

Among all the courses proposed by the Curricula 2001 Draft Report, these four seem to address most directly the curricular needs of IDAACS, as defined above. Together, they provide a strong foundation for the needs of postgraduate study and research in intelligent data acquisition and advanced computer systems. Here is a paraphrasing of these courses as they are described in the Curricula 2001 report.

Computer Architecture This course introduces students to the organization and architecture of computer systems, beginning with the standard von Neumann model and then moving forward to more recent architectural concepts. Topics include:

• Digital logic – Fundamental building blocks (logic gates, flip-flops, counters, registers, PLA); logic expressions, minimization, sum of product forms; register transfer notation; physical considerations (gate delays, fan-in, fan-out)

• Data representation: Bits, bytes, and words; numeric data representation and number bases; fixed- and floating-point systems; signed and twos-complement representations; representation of nonnumeric data (character codes, graphical data); representation of records and arrays

• Assembly level organization: Basic organization of the von Neumann machine; control unit; instruction fetch, decode, and execution; instruction sets and types (data manipulation, control, I/O); assembly/machine language programming; instruction formats; addressing modes; subroutine call and return mechanisms; I/O and interrupts

• Memory systems: Storage systems and their technology; coding, data compression, and data integrity; memory hierarchy; main memory organization and operations; latency, cycle time, bandwidth, and interleaving; cache memories (address mapping, block size, and replacement policy); virtual memory (page table, TLB); fault handling and reliability

• Interfacing and communication: I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O; interrupt structures and acknowledgment; external storage, physical organization, and drives; buses: bus protocols, arbitration, direct-memory access (DMA); introduction to networks; multimedia support; raid architectures

• Functional organization: Implementation of simple datapaths; control unit: hardwired realization vs. microprogrammed realization; instruction pipelining; introduction to instruction-level parallelism (ILP)

• Multiprocessor and alternative architectures: Introduction to SIMD, MIMD, VLIW, EPIC; systolic architecture; interconnection networks (hypercube, shuffle-exchange, mesh, crossbar); shared memory systems; cache coherence; and memory consistency

• Performance enhancement: Superscalar architecture; branch prediction; prefetching; speculative execution; multithreading; scalability

• Databases This course introduces students to the concepts and techniques of database systems. Topics include:

• History and motivation for information systems; information storage and retrieval; information management applications; information capture and representation; analysis and indexing; search, retrieval, linking, navigation; information privacy, integrity, security, and preservation; scalability, efficiency, and effectiveness

• Database systems: History and motivation; components of database systems; DBMS functions; database architecture and data independence

• Data modeling; conceptual models; object-oriented model; relational model

• Relational databases: Mapping conceptual schema to a relational schema; entity and referential integrity; relational algebra and relational calculus

• Database query languages: Overview of database languages; SQL; query optimization; 4th-generation environments; embedding non-procedural queries in a procedural language; introduction to Object Query Language

• Relational database design: Database design; functional dependency; normal forms; multivalued dependency; join dependency; representation theory

• Transaction processing: Transactions; failure and recovery; concurrency control

• Distributed databases: Distributed data storage; distributed query processing; distributed transaction model; concurrency control; homogeneous and heterogeneous solutions; client-server

• Physical database design: Storage and file

structure; indexed files; hashed files; signature files; b-trees; files with dense index; files with variable length records; database efficiency and tuning

Information Management This course introduces students to the task of organizing large volumes of information of potentially different kinds. Typically, resolution of the associated problems depends on the use of an underlying database technology, often involving networking. This course addresses the issues involved. Topics include:

• The problems associated with the management of data resources of potentially different kinds; the business perspective

• Introduction to databases; the relational model; illustrations

• Building databases: underlying methodology, database languages; particular database issues

• Information systems to serve particular purposes, including intranets and extranets; the information retrieval problem

• The design and development of information systems

• Security and control issues

• Evaluation of information systems

• Intelligent Systems This course presents both the theory and application of intelligent systems, using robotics and other control-based systems as a framework for the discussion. Topics include:

• Fundamental issues in intelligent systems: History of artificial intelligence; philosophical questions; fundamental definitions; philosophical questions; modeling the world; the role of heuristics

• Search and constraint satisfaction: Problem spaces; brute-force search; best-first search; two-player games; constraint satisfaction

• Knowledge representation and reasoning: Review of propositional and predicate logic; resolution and theorem proving; nonmonotonic inference; probabilistic reasoning; bayes theorem

• Agents: Definition of agents; successful applications and state-of-the-art agent-based systems ; agent architectures; agent theory; software agents, personal assistants, and information access; believable agents; learning agents; multi-agent systems; introduction to robotic agents; mobile agents

• Machine learning and neural networks: Definition and examples of machine learning; supervised learning; learning decision trees; learning neural networks; learning belief networks; the nearest neighbor algorithm; learning theory; the problem of overfitting; unsupervised learning; reinforcement learning

• AI planning systems: Definition and examples of planning systems; planning as search; operator-based planning; propositional planning; extending planning systems; static world planning systems; planning and execution; planning and robotics

• Robotics: Overview; configuration space; planning; sensing; robot programming; navigation and control

6. ANALYSIS AND CONCLUSIONS

The summary above provides some insights into the design of Computing Curricula 2001 and its relationship with the subject matter concerns of IDAACS. However, this summary does not pretend to be either complete or perfect. The following weaknesses are apparent.

First, there are surely topics outside the areas marked by an asterisk (*) within the core body of knowledge that are either important or even essential to the study of intelligent data acquisition and advanced computer systems. For example, the study of probability (Discrete Structures) and event-driven programming (Programming Fundamentals) would appear to be essential preparation for further study in IDAACS.

Second, there are surely may topics that lie within the subject areas marked by an asterisk (*) that are not of primary interest to IDAACS. For example, management information systems (Information Management) would appear only to be interesting to persons preparing for a career in management, rather than IDAACS.

Third, the four specific courses that were presented above as particularly useful for students interested in IDAACS do not comprise a complete list in any sense. For instance, it is very important that these students study various subjects in mathematics, including analysis, algebra, calculus, and statistics, in order to prepare well for future work in intelligent data analysis. Moreover, data analysis requires the development and interpretation of appropriate mathematical and economic models of real systems, so that a course in mathematical modeling would seem to be essential for this work.

In spite of these shortcomings, this paper has attempted to present an initial overview of the relationship between a modern curriculum design and the field if intelligent data analysis and advanced computer systems. Future efforts to design a more complete curriculum model for IDAACS may take into account the information in this discussion, as well as other information in the Curricula 2001 report itself and other documented sources. Since the Curricula 2001 report is not yet in its final form, interested readers should consult the Web site

www.computer.org for up-to-date information about the current version of that report.

7. REFERENCES

[1] Computing Curricula 2001 Draft Report. ACM/IEEE Joint Curriculum Task Force (February 1, 2001). www.computer.org

[2] A. Tucker (ed), B. Barnes, R. Aiken, K. Barker, K. Bruce, J. Cain, S. Conry, G. Engel, R. Epstein, D. Lidtke, M. Mulder, J. Rogers, E. Spafford, and A. Turner, Computing Curricula 1991. ACM/IEEE Joint Curriculum Task Force (1991).

[3] ACM Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the undergraduate program in computer science. Communications of the ACM, 22(3):147-166, March 1979.

[4] ACM Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the undergraduate program in computer science. Communications of the ACM, 11(3):151-197, March 1968.

Allen B. Tucker is the Anne T. and Robert M. Bass Professor of Natural Sciences in the Department of Computer Science at Bowdoin College, where he has taught since 1988. He held similar positions at Colgate and Georgetown Universities. He earned a BA in mathematics from Wesleyan University and an MS and PhD in computer science from Northwestern University.

Professor Tucker is the author or coauthor of several books and other publications in the areas of programming languages, natural language processing, and computer science education, which are his current research areas. He has given many talks, panel discussions, and workshop presentations in these areas, and has served as a reviewer for various journals, NSF programs, and curriculum projects. He has also served as a consultant to several colleges, universities, and other institutions in the areas of computer science curriculum, software design, programming languages, and natural language processing applications.

Professor Tucker co-chaired the ACM/IEEE-CS Joint Curriculum Task Force that developed Computing Curricula 1991, for which he received the ACM's Outstanding Contribution Award and shared the IEEE's Meritorious Service Award. He is a co-author of the 1986 Liberal Arts Model Curriculum in Computer Science and Editor-in-Chief of the 1997 CRC Handbook of Computer Science and Engineering. He received the ACM SIGCSE award for Outstanding Contributions to Computer Science Education in February 2001. In Spring 2001, he was a Fulbright Lecturer at the Ternopil Academy of National Economy (TANE) in Ukraine. He is a Fellow of the ACM and has been a member of the ACM, the IEEE Computer Society, Computer Professionals for Social Responsibility, and the Liberal Arts Computer Science (LACS) Consortium.