

Computing Curricula: Computer Engineering

Murali R. Varanasi
(Presented by James H. Aylor)

IEEE Computer Society / ACM
Computing Curricula – Computer Engineering
Task Force

Microelectronics Systems Education Conference (MSE'03), June 1 – June 2, 2003



Outline



- Overview of the CC2001 project
- Computing Curricula Computer Science & Computing Curricula Software Engineering
- Computing Curricula Computer Engineering
- The Computer Engineering Body of Knowledge
- Implementing CPE Curricula
- Completion of the CCCE Task Force Work
- Questions and Plea for Reviewers



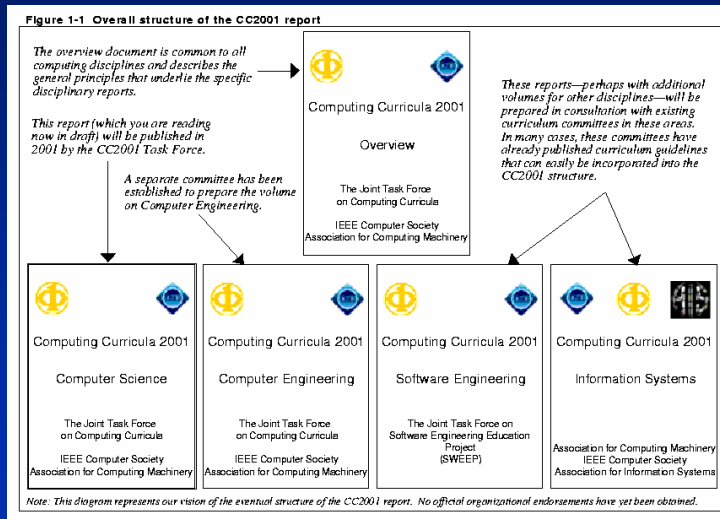
Background: CC2001

• Computing Curricula 2001

- Joint effort of IEEE-CS and ACM
- Started in 1998
- Revise and update 1991 model curricula
- Address developments of past decade and endure through the next decade
- Separate volumes for Computer Science, Computer Engineering, Software Engineering, Information Systems
- Computer Science Volume completed in Dec. 2001
<http://www.computer.org/education/cc2001/>



Structure of the CC2001 Report





Computing Curricula – Computer Science (CCCS) Volume Outline



- Chapter 1. Introduction
- Chapter 2. Lessons from past reports
- Chapter 3. Changes in the computer science discipline
- Chapter 4. Principles
- Chapter 5. Overview of the CS Body of Knowledge
- Chapter 6. Overview of the Curricular Models
- Chapter 7. Introductory Courses
- Chapter 8. Intermediate Courses
- Chapter 9. Completing the Curriculum



CCCS Volume Outline (continued)



- Chapter 10. Professional Practice
- Chapter 11. Characteristics of CS Graduates
- Chapter 12. Computing Across the Curriculum
- Chapter 13. Strategy and Tactics
- Acknowledgements
- Bibliography
- Appendix A – The CS Body of Knowledge
- Appendix B – CS Course Descriptions



Computing Curricula - Software Engineering (CCSE)



- 1998 - ACM & IEEE-CS Software Engineering Education Project (SWEEP) published draft accreditation guidelines for undergraduate software engineering curricula (IEEE-CS “Computer”)
- Spring 2001 - SWEEP began work on the Computing Curriculum - Software Engineering (CCSE) Volume
- Fall 2001 - SWEEP replaced by joint ACM & IEEE-CS CCSE Steering Committee (multi-national)
- Spring 2002 - SEEK structure & knowledge areas defined



CCSE Accomplishments (highlights)



- Summer 2002
 - Comprehensive review of SEEK draft (leading SWE educators, practitioners, researchers)
 - SEEK draft released in August for public review.
- Fall 2002
 - Pedagogy focus group established a process and work plan
 - SEEK modified and 2nd version posted (December)



CCSE Volume Outline



- Objectives and Guiding Principles of CCSE
 - CCSE Principles
 - Curriculum Outcomes
- Process of Determining the SEEK
- Knowledge Areas, Units, and Topics
 - Core material
 - Unit of time
- Relationship of the SEEK to Curriculum
- SE Education Knowledge Areas



Computing Curricula: Computer Engineering



- **Computer Engineering Task Force**
 - Established in 2001
 - 17 members currently
 - Representation of diverse interests/institutions
- **Charge of the CCCE Task Force**
 - Define characteristics of CpE graduates
 - Define the CpE body of knowledge
 - Define CpE core requirements
 - Define engineering practice issues for CpE curric.
 - Provide sample curriculum implementations



Computing Curricula: Computer Engineering



- Distinguish CpE vs. EE & CS & SWE
- CpE Curricular Guidelines (beyond the body of knowledge):
 - Science & math, engineering core
 - Intersections with EE and CS
 - Emphasize design and creativity
 - Laboratory experience
 - Industry-standard modern tools



CCCE Task Force



Dave Soldan, Chair

Vic Nelson	Bob Sloan
Mitch Theys	Murali Varanasi
Pradip Srimani	Jim Aylor
John Impagliazzo	Gerald Engel
Andrew McGettrick	Ron Hoelzeman
Danial J. Neebel	Esther A. Hughes
Joe Hughes	Bob Klenke
Alan Clements	Douglas Lyon



CCCE Task Force Activities



- February 2001 – Task Force Organized to develop CCCE Volume
- Summer 2001 – CCCE Volume outline developed and body of knowledge (BOK) areas defined
- Fall 2001 – Task force members and other experts wrote initial BOK areas:
 - One task force member wrote initial BOK area
 - 2nd task force member reviewed the BOK area
 - Experts solicited to review the BOK area



CCCE Task Force Activities



- May 2002 – Full Task Force reviewed the BOK and published draft CCCE BOK on the web for public review
- November 2002 – NSF-sponsored workshop brought experts together to review the CCCE Body of Knowledge
- February 2003 – First draft of the full CCCE report published on the web for review (minus sample curriculum models)



Outline of the CCCE Draft Volume



1. Introduction
2. Computer Engineering Principles
3. Characteristics of Computer Engineering Graduates
4. Overview of the Computer Engineering Body of Knowledge
5. Integration of Engineering Practice Into the Computer Engineering Curriculum
6. Professionalism and Computer Engineering



Outline of the CCCE Draft Volume



7. Curriculum Implementation Issues
8. Organizational Issues
9. Summary and Conclusions

Appendix A: Body of Knowledge

Appendix B: Sample Curriculum
Implementations



What is Computer Engineering ?



- Computer engineering embodies the science and the technology of design, construction, implementation and maintenance of the hardware and the software components of modern computing systems and computer-controlled equipment.
- Computer engineers are solidly grounded in the theories and principles of computing, mathematics and engineering, and apply these theoretical principles to design hardware, software, networks, and computerized equipment and instruments to solve technical problems in diverse application domains.



Basic Principles



- Computer engineering draws its foundations from a wide variety of disciplines.
- CCCE should seek to identify the fundamental skills and knowledge that **ALL** computer engineering students must possess.
- The required body of knowledge must be made as small as possible.
- The development of CCCE must be broadly based.
- CCCE must include an appropriate and necessary laboratory experience component.
- CCCE core must acknowledge that engineering curricula should be accredited.



Basic Principles (2)



- Computing is a broad field that extends well beyond the boundaries of computer engineering.
- The rapid evolution of computer engineering requires an ongoing review of the corresponding curriculum.
- Development of a computer engineering curriculum must be sensitive to changes in technology, new developments in pedagogy, and the importance of lifelong learning.
- CCCE must go beyond knowledge units to offer significant guidance in terms of individual course design.



Basic Principles (3)



- CC2001 must strive to be international in scope.
- CC2001 must include professional practice as an integral component of the undergraduate curriculum.
- CC2001 must include discussions of strategies and tactics for implementation along with high-level recommendations.



Defining a Body of Knowledge



- **Hierarchical Structure**
 - Disciplinary subfields (e.g., digital logic)
 - Units or thematic modules (e.g., switching theory)
 - Topics (e.g., number systems)
- **Core vs. Elective Units**
 - Core => should be included in all programs
 - Elective => inclusion based on program objectives and/or student interest
- ***Organization of BOK does not imply organization of a curriculum!***



CCCE Body of Knowledge Topic Areas (1)



- **Areas Likely to Contain Core Topics**
 - SPR - Social and Professional Issues
 - CSE - Computer Systems Engineering
 - CAO - Computer Architecture and Organization
 - SWE - Software Engineering
 - ESY - Embedded Systems
 - OPS - Operating Systems
 - CSY - Circuits and Systems
 - NWK - Networks
 - ELE - Electronics



CCCE Body of Knowledge Topic Areas (2)



- **Areas Likely to Contain Core Topics (cont'd)**
 - DIG - Digital Logic
 - PRF - Programming Fundamentals
 - ALG - Algorithms and Complexity
 - DSC - Discrete Structures
- **Areas Likely to be Mostly Elective**
 - DSP - Digital Signal Processing & Multimedia
 - VLS - VLSI/ASIC Design
 - ACP - Alternative Computing Paradigms
 - TFT - Testing and Fault Tolerance
 - DVS - Digital System Verification



CCCE – Core Hours (Lecture/Contact Hours)



36 - Discrete Structures	73 - Circuits, Signals, Systems
44 - Programming Fundamentals	45 - Electronics
34 - Algorithms & Complexity	50 - Digital Logic
45 - Operating Systems	85 - Computer Arch. & Organization
11 - Software Engineering	12 - Computer Systems Engineering
16 - Social/Professional	22 - Embedded Systems
24 - Computer Networks	

Total = 497 (debate continuing) – 33 semester “credit hours”
Goal about ¼ of typical curriculum



Computer Architecture and Organization (CAO)



- CAO0. History and overview of computer architecture [core]
- CAO1. Fundamentals of computer architecture [core]
- CAO2. Computer arithmetic [core]
- CAO3. Memory system organization and architecture [core]
- CAO4. Interfacing and communication [core]
- CAO5. Interface subsystems [core]
- CAO6. Processor systems design [core]
- CAO7. Organization of the CPU [core]
- CAO8. Performance [core]
- CAO9. Performance enhancements [elective]
- CAO10. Multiprocessing [core]



CAO1. Fundamentals of Computer Architecture [core]



- Minimum core coverage time: 18 hours
- Topics:
 - Organization of the von Neumann machine
 - Instruction formats
 - The fetch/execute cycle; instruction decoding and execution
 - Registers and register files.
 - Instruction types and addressing modes
 - Subroutine call and return mechanisms
 - Programming in assembly language
 - I/O techniques and interrupts
 - Other design issues.



CAO1. Fundamentals of Computer Architecture



- **Learning Objectives - CPE Graduates will be able to...**
 - explain the organization of a von Neumann machine and its major functional units.
 - explain how an instruction is fetched from memory and executed.
 - articulate the strengths and weaknesses of the von Neumann architecture.
 - explain the relationship between the representation of machine level operation at the binary level and their representation by a symbolic assembler.
 - explain why a designer adopted a given different instruction formats, such as the number of addresses per instruction and variable length vs. fixed length formats.
 - write small programs and fragments of assembly language code to demonstrate an understanding of machine level operations.



CAO9. Performance Enhancements [elective]



- **Minimum core coverage time: 0 hours**
- **Topics:**
 - Superscalar architecture
 - Branch prediction
 - Prefetching
 - Speculative execution
 - Multithreading
 - Scalability
 - Short vector instruction sets; Streaming extensions, Altivec; relationship between computer architecture and multimedia applications.



CAO9. Performance Enhancements



- **Learning objectives:**
 - Discuss how various architectural enhancements affect system performance.
 - Discuss how parallel processing approaches can be applied to the design of scalar and superscalar processors.
 - Discuss how vector processing techniques can be applied to enhance instruction sets to be used for multimedia, signal processing, etc.
 - Appreciate how each of the functional parts of a computer system affect its overall performance. Be able to estimate the effect on system performance of changes to functional units.



Engineering Practice in the CPE Curriculum



- **Simply covering the CpE body of knowledge is not enough!**
- **Engineering design throughout the curriculum**
 - Incorporate standards and realistic constraints
 - Consider the “-abilities” (manufacturability, testability, etc.)
 - Consider other issues: economic, social, ethical, etc.
- **Meaningful laboratory experience**
- **Proficiency with modern design and analysis tools**
- **Emphasis on developing communication skills**
- **Teamwork skills & experiences**



Industry Needs – Top Ten Factors in 2001



National Association of Colleges and Employers (NACE) annual survey to determine what qualities employers consider most important in applicants seeking employment.

1. Communication skills (verbal and written)
2. Honesty/integrity
3. Teamwork skills
4. Interpersonal skills
5. Motivation/initiative
6. Strong work ethic
7. Analytical skills
8. Flexibility & adaptability
9. Computer skills
10. Self-confidence



Professionalism



- Computer engineers must consider the professional, societal, and ethical context in which they practice
- Computer engineers design and implement computing systems that affect the public
- Computer engineers should hold a special sense of responsibility
- Almost every aspect of their work can have a public consequence
- Consequences of professional practice should focus on the public good



Completing the CpE Curriculum



- Mathematics
 - CpEs should have a solid foundation in math, especially integral and differential calculus and probability & statistics
 - CpE courses should require application of math
 - Some CpE courses should incorporate advanced math (linear algebra, differential equations, etc.)
- Basic Science
 - CpEs should have a solid foundation in basic science
 - Many CpE topics based on principles from physics
 - Some CpE courses may build on chemistry, biology, etc.
- Humanities, Arts, etc.
 - Engineers should be good citizens
 - Engineers must understand how their work impacts society



Papers/Presentations on the Work of the CCCE Task Force



- 2001 FIE Panel
- 2002 ASEE Panel
- 2002 FIE Panel
- 2002 NSF Workshop on the CCCE BOK
- 2003 SigCSE Panel
- 2003 ECEDHA Annual Meeting Panel



CCCE - What Happens Next ?

- **Current draft of CCCE report available at:**
<http://www.eng.auburn.edu/ece/CCCE/>
 - Public review and comment via website wanted
- **Summer 2003 – Incorporation of review comments & sample curricula**
 - MORE review and comment via website wanted
- **Fall 2003 - Full review of report**
 - Wide participation sought, including international
- **Fall 2003 - Final draft published on web and submitted to IEEE-CS and ACM**



Discussion





Codes of Ethics and Practices – “Do The Right Thing”



- National Society of Professional Engineers - *NSPE Code of Ethics for Engineers*
- Institute of Electrical and Electronic Engineers (IEEE): *IEEE Code of Ethics*
- Association for Computing Machinery (ACM): *ACM Code of Ethics and Professional Conduct*
- ACM/IEEE-Computer Society: *Software Engineering Code of Ethics and Professional Practice*
- International Federation for Information Processing (IFIP): *Harmonization of Professional Standards and also Ethics of Computing*
- Association of Information Technology Professionals (AITP): *AITP Code of Ethics and the AITP Standards of Conduct*



Social and Professional Issues (SPR)



- SPR0. History and overview of social and professional issues [core]**
- SPR1. Social context of computing [core]**
- SPR2. Methods and tools of analysis [core]**
- SPR3. Professional and ethical responsibilities [core]**
- SPR4. Risks and liabilities of computer-based systems [core]**
- SPR5. Intellectual property [core]**
- SPR6. Privacy and civil liberties [core]**
- SPR7. Computer crime [elective]**
- SPR8. Economic issues in computing [elective]**
- SPR9. Philosophical frameworks [elective]**



Computer Systems Engineering (CSE)



- CSE 0. History and overview of computer systems design [core]
- CSE 1. Overview of systems engineering [core]
- CSE 2. Theoretical considerations [elective]
- CSE 3. Life cycle [core]
- CSE 4. Requirements analysis and elicitation [core]
- CSE 5. Specification [core]
- CSE 6. Architectural design [core]
- CSE 7. Implementation
- CSE 8. Testing
- CSE 9. Maintenance
- CSE 10. Project management
- CSE 11. Specialist systems
- CSE 12. Hardware and software co-design



Computer Architecture and Organization (CAO)



- CAO0. History and overview of computer architecture [core]
- CAO1. Fundamentals of computer architecture [core]
- CAO2. Computer arithmetic [core]
- CAO3. Memory system organization and architecture [core]
- CAO4. Interfacing and communication [core]
- CAO5. Interface subsystems [core]
- CAO6. Processor systems design [core]
- CAO7. Organization of the CPU [core]
- CAO8. Performance [core]
- CAO9. Performance enhancements [elective]
- CAO10. Multiprocessing [core]



Software Engineering (SWE)



- SWE0. History and overview of software engineering [core]
- SWE1. Software processes [core]
- SWE2. Software requirements and specifications [core]
- SWE3. Software design [core]
- SWE4. Software testing and validation [core]
- SWE5. Software evolution [elective]
- SWE6. Software tools and environments [core]
- SWE7. Software project management [elective]



Embedded Systems (ESY)



- ESY0. History and overview of embedded systems [core]
- ESY1. Fundamentals of embedded systems [core]
- ESY2. Language issues [core]
- ESY3. Hardware considerations [core]
- ESY4. Mapping between languages and hardware [core]
- ESY5. Real-time Operating Systems [elective]
- ESY6. Classification of embedded systems [elective]
- ESY7. Software engineering considerations [elective]
- ESY8. Particular techniques and applications [elective]



Operating Systems (OPS)



- OPS0. History and overview of operating systems [core]
- OPS1. Operating system function and design [core]
- OPS2. Operating system principles [core]
- OPS3. Concurrency [core]
- OPS4. Scheduling and dispatch [core]
- OPS5. Memory management [core]
- OPS6. Device management [core]
- OPS7. Security and protection [elective]
- OPS8. File systems [elective]
- OPS9. System performance evaluation [core]



Circuits and Systems (CSY)



- CSY0. History and overview of systems and circuits [core]
- CSY1. Fundamental Electrical Quantities (core)
- CSY2. Resistive Circuits and Networks (core)
- CSY3. Reactive Circuits and Networks (core)
- CSY4. Frequency Response (core)
- CSY5. Sinusoidal Analysis (core)
- CSY6. Convolution (core)
- CSY7. Discrete Time Signals (core)
- CSY8. Fourier Analysis (core)
- CSY9. Filters (elective)
- CSY10. Laplace Transforms (elective)
- CSY11. z – Transforms (elective)
- CSY12. Digital Filters (elective)



Networks (NWK)



- NWK0. History and overview of networks [core]**
- NWK1. Communications Network Architecture [core]**
- NWK2. Communications Network Protocols [core]**
- NWK3. Local and Wide Area Networks [core]**
- NWK4. The web as an example of client-server computing [core]**
- NWK5. Data Security and Integrity [elective]**
- NWK6. Performance Evaluation [elective]**
- NWK7. Data Communications [elective]**
- NWK8. Wireless and mobile computing [elective]**

IEEE CS/ACM Computing Curricula - Computer Engineering

45



Electronics (ELE)



- ELE0. History and overview of electronics [core]**
 - ELE1. Electronic properties of materials [core]**
 - ELE2. Diodes and diode circuits [core]**
 - ELE3. MOS transistors and biasing [core]**
 - ELE4. MOS logic families [core]**
 - ELE5. Bipolar transistors and logic families [core]**
 - ELE6. Design parameters and issues [core]**
 - ELE7. Storage elements [core]**
 - ELE8. Interfacing logic families and standard buses [core]**
 - ELE9. Operational amplifiers [core]**
 - ELE10. Data conversion circuits [core]**
- (continued)

IEEE CS/ACM Computing Curricula - Computer Engineering

46



Electronics (continued)



- ELE11. SPICE circuit simulation [core]
- ELE12. Electronic voltage and current sources [elective]
- ELE13. Linear amplification and biasing [elective]
- ELE14. Single-transistor amplifiers [elective]
- ELE15. Multistage transistor amplifiers [elective]
- ELE16. Power circuits [elective]
- ELE17. Feedback in electronics [elective]
- ELE18. Active filters [elective]
- ELE19. Integrated circuit building blocks [elective]
- ELE20. Circuits for wireless applications [elective]



Digital Logic (DIG)



- DIG0. History and overview of digital logic [core]
- DIG1. Switching theory [core]
- DIG2. Combinational logic circuits [core]
- DIG3. Modular design of combinational circuits [core]
- DIG4. Memory elements [core]
- DIG5. Sequential logic circuits [core]
- DIG6. Register Transfer Logic [core]
- DIG7. Digital Systems Design [core]



Programming Fundamentals (PRF)



- PRF0. History and overview of programming fundamentals [core]
- PRF1. Fundamental programming constructs [core]
- PRF2. Algorithms and problem-solving [core]
- PRF3. Fundamental data structures [core]
- PRF4. Programming Paradigms [core]
- PRF5: Recursion [core]
- PRF6. Object-oriented programming
- PRF7. Event-driven and concurrent programming
- PRF8. Using APIs



Algorithms and Complexity (ALG)



- ALG0. History and overview of algorithms and complexity [core]
- ALG1. Basic algorithmic analysis [core]
- ALG2. Algorithmic strategies [core]
- ALG3. Fundamental computing algorithms [core]
- ALG4. Distributed algorithms [core]
- ALG5. Basic computability theory [core]
- ALG6. The complexity classes P and NP [elective]



Discrete Structures (DSC)



- DSC0. History and overview of discrete structures
- DSC1. Functions, relations, and sets [core]
- DSC2. Basic logic [core]
- DSC3. Proof techniques [core]
- DSC4. Basics of counting [core]
- DSC5. Graphs and trees [core]
- DSC6. Discrete probability [core]
- DSC7. Recursion [elective]



DIGITAL SIGNAL PROCESSING & MULTIMEDIA (DSP)



- DSP1. Overview of Digital Audio and its application
- DSP2. Discrete Time Signals (SC9)
- DSP3. Multimedia programming, data streaming
- DSP4. Wave Table Generation
- DSP5. Convolution (SC8)
- DSP6. Fourier Analysis (SC10, modified)
- DSP7. Audio Processing
- DSP8. Generalized Modulations and Demodulations
- DSP9. LaPlace Transforms (SC12 with modifications)
- DSP10. Z - transforms (SC13 with modifications)
- DSP11. Digital Filters (SC14, with modifications)
- DSP12. Digital Image Processing Fundamentals

(continued)



DIGITAL SIGNAL PROCESSING & MULTIMEDIA (DSP)



- DSP13. Simple Graphics**
- DSP14. Displaying Images**
- DSP15. Reading and Writing Image Files**
- DSP16. Edge Detection**
- DSP17. Boundary Processing**
- DSP18. Image Enhancement Techniques**
- DSP19. Achromatic and Colored Light**
- DSP20. Thresholding techniques**
- DSP21. Morphological filtering**
- DSP22. Warping**
- DSP23. The Cosine Transform**
- DSP24. The InLine MPEG CODEC**
- DSP25. The Wavelet Transform**



VLSI and ASIC Design (VLS)



- VLS1. MOS Transistor Fundamentals**
- VLS2. Processing and Layout**
- VLS3. Function of the Basic Inverter Structure**
- VLS4. Circuit Characterization and Performance**
- VLS5. Combinational Logic Circuits**
- VLS6. Sequential Logic Circuits**
- VLS7. Alternative Circuit Structures/Low Power Design**
- VLS8. Semiconductor Memories and Array Structures**
- VLS9. Chip Input/Output Circuits**
- VLS10. Semi custom Design Technologies**
- VLS11. ASIC Design Methodology**



Alternative Computing Paradigms (ACP)



- ACP1. Overview/History [core]
- ACP2. Paradigms [core]
- ACP3. Architectures
- ACP4. Operating systems issues
- ACP5. Software issues
- ACP6. Algorithms
- ACP7. Applications



Testing and Fault Tolerance (TFT)



- TFT1 – Faults and Fault Models in Digital Circuits
- TFT2 - Test generation methods
- TFT3 - Design for testability
- TFT4 - Testing non-stuck-at faults
- TFT5 - System-level test and diagnosis
- TFT6 - Reliability and fault tolerance definitions
- TFT7 - Error detecting and correcting codes
- TFT8 – Fault Tolerant System Design
- TFT9 – Software approaches and software fault tolerance



Digital System Verification (DSV)



DSV0. History and Overview Including Pentium Bugs and other Horror Stories, Verification vs. Validation. Relationship of Good Design Practice to Verification (3 hours)

DSV2. Comparison of Simulation, Testing, and Formal Verification (Timing Analysis) for Timing (4 hours)

DSV3. Formal Verification: Model Checking (10 hours)

DSV4: Formal Verification: Proofs (6 hours)

DSV5: Formal Verification: Equivalence Checking (3 hours)

DSV6: Verification by Simulation and Testbenches (4 hours)

DSV7: Verification by Assertions and Verification Languages (4 hours)

DSV8: Verification by Testing (2 hours)

DSV9: Other Verification: Signal Integrity, Specification, Reliability, Safety, Power, Cooling, ASIC Physical Design, ... (3 hours)

DSV10: Comparison and Contrast of Verification, Testing, and Reliability (1 hour)

DSV11: Configuration Control, Bug Tracking, Regression Testing (2 hours)

DSV12: Economics of Verification (2 hours)

IEEE CS/ACM Computing Curricula - Computer Engineering

57



Digital Systems Engineering: Signal Integrity (DSI)



DSI0: History and Overview, Motivation, Importance, Horror Stories

DSI1: Signals

DSI2: Lossless Transmission Lines

DSI3: Coupled Lines

DSI4: Measurement

DSI5: Simulation

DSI6: Signaling

DSI7: Power distribution

DSI8: EMI/EMC(?)

IEEE CS/ACM Computing Curricula - Computer Engineering

58



Intelligent Systems and Automation (ISA)



- Yet to be defined



CSE 3: Life Cycle [core]



Minimum core coverage : 2 hours

Topics:

1. Nature of life cycle, role of life cycle model. Quality in relation to the life cycle.
2. Influence of system size on choice of life cycle model and nature of system – agility issues.
3. Different models of the life cycle – strengths and weaknesses of each.
4. The concept of process. Process improvement. Basis for this is information.
5. Gathering information.
6. Maturity models. Standards and guidelines.

Learning objectives:

1. Recognize the need for a disciplined approach to system development and explain the elements of this in particular contexts.
2. Explain how data should be gathered to inform process improvement.



- Please send comments to

cpe_com@computer.org