IT CURRICULUM

Alan Fekete (Uni of Sydney) representing Lin Padgham and CORE

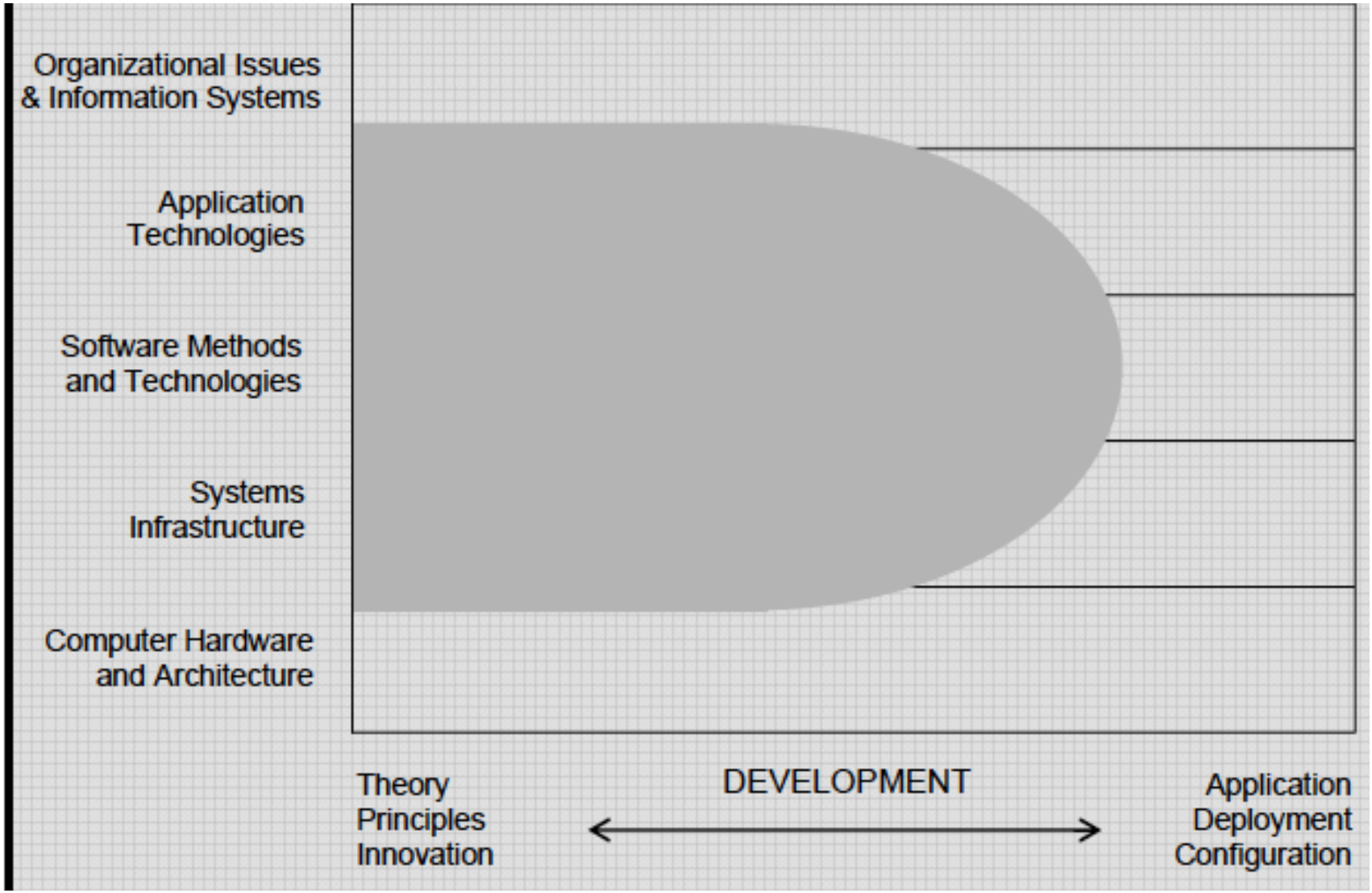
Summary

- The term “Computer Science” should mean something
  - Degree title, or name of major
- The definition of “Computer Science” graduate should follow ACM/IEEE approach
  - See CC2013 (latest instantiation)
  - At https://www.acm.org/education/curricula-recommendations
- CORE does not constrain what unis might offer as other types of computing education
  - But we do believe that the CS variety has real value, and should be available widely

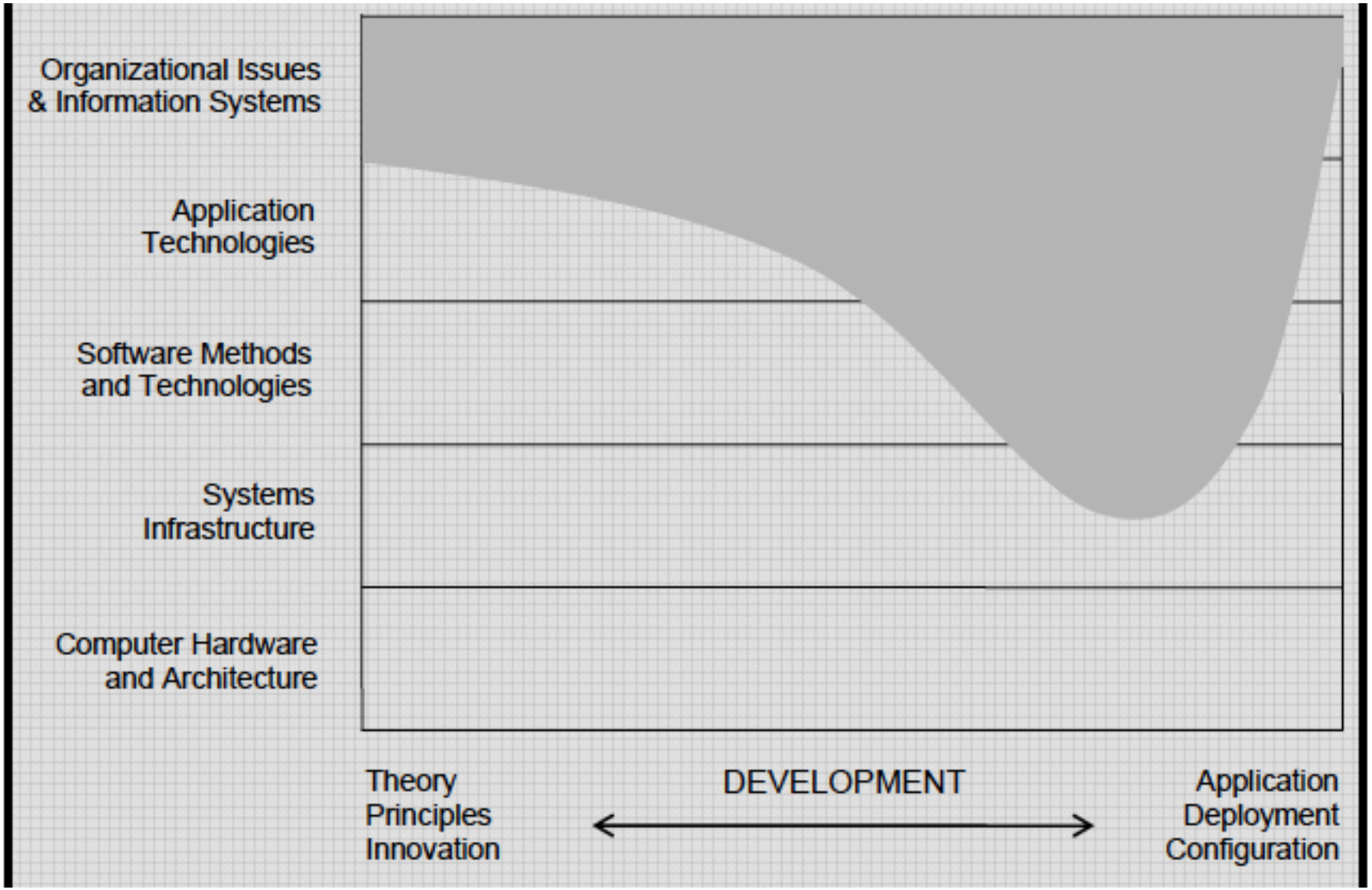
Computer Science (ACM description)

- Computer science spans a wide range, from its theoretical and algorithmic foundations to cutting-edge developments in robotics, computer vision, intelligent systems, bioinformatics, and other exciting areas.
- We can think of the work of computer scientists as falling into three categories.
  - They design and implement software. Computer scientists take on challenging programming jobs. They also supervise other programmers, keeping them aware of new approaches.
  - They devise new ways to use computers. Progress in the CS areas of networking, database, and human-computer-interface enabled the development of the World Wide Web. Now CS researchers are working with scientists from other fields to make robots become practical and intelligent aides, to use databases to create new knowledge, and to use computers to help decipher the secrets of our DNA.
  - They develop effective ways to solve computing problems. For example, computer scientists develop the best possible ways to store information in databases, send data over networks, and display complex images. Their theoretical background allows them to determine the best performance possible, and their study of algorithms helps them to develop new approaches that provide better performance.
- Computer science spans the range from theory through programming. Curricula that reflect this breadth are sometimes criticized for failing to prepare graduates for specific jobs. While other disciplines may produce graduates with more immediately relevant job-related skills, computer science offers a comprehensive foundation that permits graduates to adapt to new technologies and new idea
- From CC2005 Overview Report

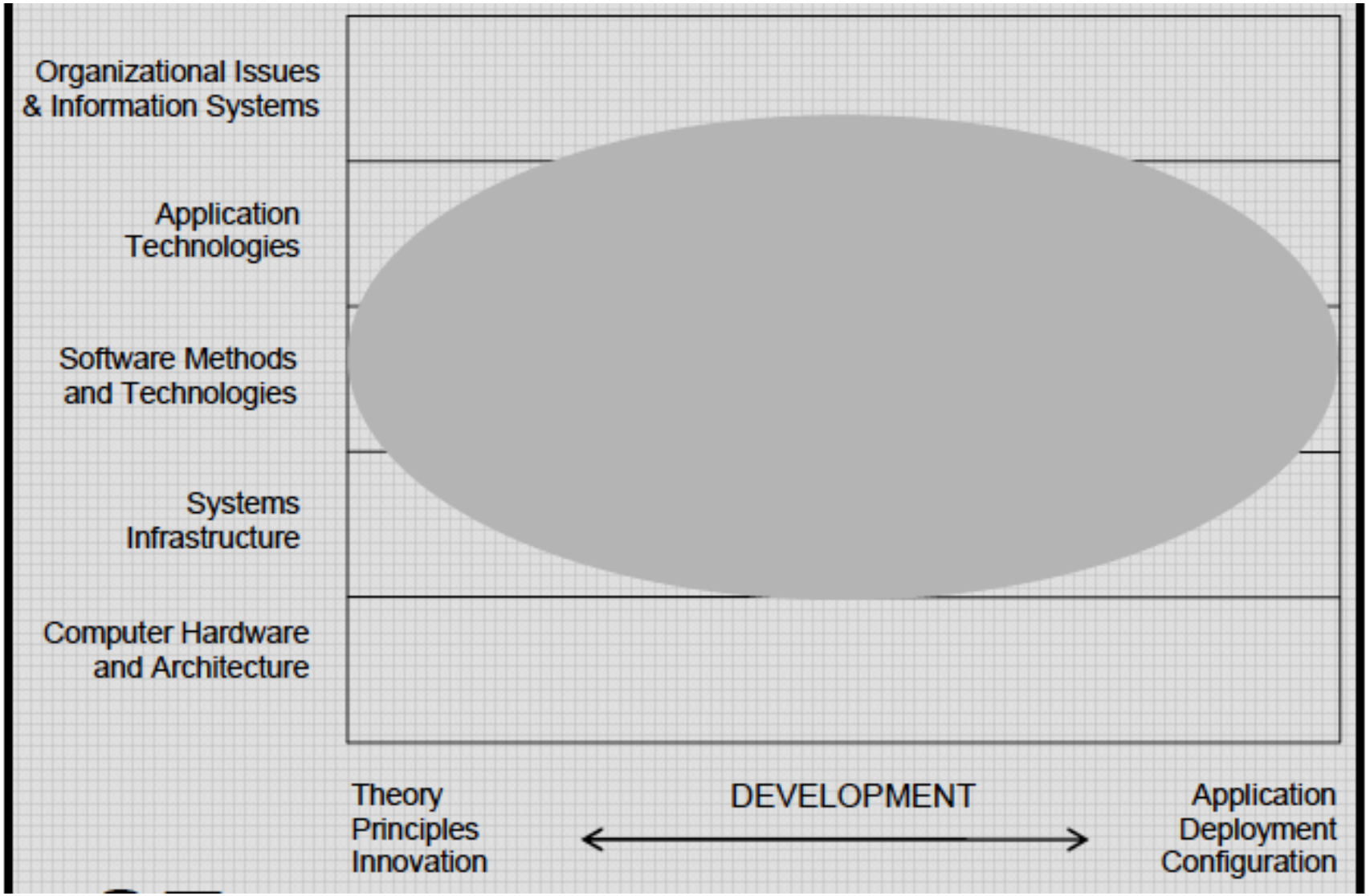
Computer Science (ACM diagram)



Compare with IS (ACM diagram)



Compare with SE (ACM diagram)



CORE accepts ACM/IEEE CS curriculum

- Large community-wide effort to define this
  - Including international consultation, industry advice
  - Leaders in 2013: Mehram Sahami (Stanford), Steve Roach (Exelis Inc)
- Kept fairly uptodate (revisions CS2001, CS2008, CS2013)
- Suited to major in US liberal-arts bachelors degree
  - 4 year degree, but also has a substantial breadth requirement
  - So the space for the major (about 8-10 semester subjects) is reasonable as minimum in Australian 3-year Bachelors
- It is not feasible for Australia to deviate much from international understanding, nor worth the huge effort to develop our own documents

Structure of ACM CS2013

- Content defined in knowledge areas (KAs)
- KAs do not need to match with subjects in curriculum
  - A subject may mix parts of several KAs
  - One KA may be spread among several subjects
- In each knowledge area, some topics are "Tier-1" (required for any graduate)
  - Others are "Tier-2" (80% of these must be covered)
  - Others are elective (coverage at discretion of the program, and/or choice of the student)
- Outcomes defined as Familiarity, Usage, or Assessment
  - (similar to simplified Bloom taxonomy)

Lecture-hours (start of table)
(1 semester subject = 3 credit hours)

Knowledge Area	CS2013	
	Tier1	Tier2
AL-Algorithms and Complexity	19	9
AR-Architecture and Organization	0	16
CN-Computational Science	1	0
DS-Discrete Structures	37	4
GV-Graphics and Visualization	2	1
HCI-Human-Computer Interaction	4	4
IAS-Information Assurance and Security	3	6
IM-Information Management	1	9
IS-Intelligent Systems	0	10

Lecture-hours (rest of table)

NC-Networking and Communication	3	7
OS-Operating Systems	4	11
PBD-Platform-based Development	0	0
PD-Parallel and Distributed Computing	5	10
PL-Programming Languages	8	20
SDF-Software Development Fundamentals	43	0
SE-Software Engineering	6	22
SF-Systems Fundamentals	18	9
SP-Social Issues and Professional Practice	11	5
Total Core Hours	**165**	**143**

Example from AL/Fundamental Data Structures and Algorithms

[Core-Tier1]

- Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, and mode in a list, approximating the square root of a number, or finding the greatest common divisor
- Sequential and binary search algorithms
- Worst case quadratic sorting algorithms (selection, insertion)
- Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)
- Hash tables, including strategies for avoiding and resolving collisions
- Binary search trees
  - Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
- Graphs and graph algorithms
  - Representations of graphs (e.g., adjacency list, adjacency matrix)
  - Depth- and breadth-first traversals

[Core-Tier2]

- Heaps
- Graphs and graph algorithms
  - Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
  - Minimum spanning tree (Prim's and Kruskal's algorithms)
- Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)

Example continued

Learning Outcomes:

[Core-Tier1]

1. Implement basic numerical algorithms. [Usage]
2. Implement simple search algorithms and explain the differences in their time complexities. [Assessment]
3. Be able to implement common quadratic and O(N log N) sorting algorithms. [Usage]
4. Describe the implementation of hash tables, including collision avoidance and resolution. [Familiarity]
5. Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing. [Familiarity]
6. Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data. [Familiarity]
7. Explain how tree balance affects the efficiency of various binary search tree operations. [Familiarity]
8. Solve problems using fundamental graph algorithms, including depth-first and breadth-first search. [Usage]

9. Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context. [Assessment]

Graduate characteristics

- Technical understanding of computer science
- Familiarity with common themes and principles
- Appreciation of the interplay between theory and practice
- System-level perspective
- Problem-solving skills
- Project experience
- Commitment to life-long learning
- Commitment to professional responsibility
- Communication and organizational skills
- Awareness of the broad applicability of computing
- Appreciation of domain-specific knowledge

Some of the challenges in a CS curriculum

- Not many jobs that actually call for “CS”; most graduates work as software developers
  - How to balance teaching principles that last vs current job-needed details
    - Eg relational algebra or Oracle 11 query syntax
  - Will employers be willing to wait during catchup training?
- How can we teach skills in transfer and in connecting theory and practice?
- Mathematical background (or lack of it!) among students
- Shared subjects among CS and other computing degrees
  - Eg different approach to topics expected within one classroom
- Attracting students to CS in competition with more immediately vocational degrees