Computing Curricula 2004

Overview Report

including

A Guide to Undergraduate Degree Programs in Computing

for undergraduate degree programs in

Computer Engineering
Computer Science
Information Systems
Information Technology
Software Engineering

Joint Task Force for Computing Curricula 2004

A cooperative project of

The Association for Computing (ACM)
The Association for Information Systems (AIS)
The Computer Society (IEEE-CS)

STRAWMAN DRAFT
1 June 2004

Joint Task Force for Computing Curricula 2004

	Affiliation	CC Representative
Russell Shackelford (chair) # + * *Stanford University*	ACM	CS2001
Lillian (Boots) Cassel *Villanova University*	CAC/CSAB	---
James Cross * *Auburn University*	IEEE-CS	CS2001
Gordon Davies + * *Consultant*	BCS, ACM	CS2001
John Impagliazzo # + * *Hofstra University*	ACM	CE2005
Reza Kamali * *Purdue University (Calumet)*	ACM-SIGITE	IT2005
Eydie Lawson *Rochester Inst. of Technology*	ACM-SIGITE	IT2005
Richard LeBlanc *Georgia Tech*	ACM	SE2004
Andrew McGettrick # + * *University of Stratclyde*	BCS, ACM	CS2001/SE2004/CE2004
Robert Sloan * *University of Illinois (Chicago)*	IEEE-CS	CS2001/CE2004
Heikki Topi * *Bentley College*	AIS	IS2002
Martin van Veen *Open University of the Netherlands*	IFIP	---

Legend:

# Author

+ Editor

* Member of the Focus Group

Contents

Summary

Computing has become fundamental to the education of those who will participate in modern society. It provides the infrastructure by which we communicate, do our work, conduct our business, and manage our affairs. Computing has dramatically influenced progress in science, engineering, business, and other avenues of human endeavor. In modern times, nearly everyone needs to use computers, and many will want to study computing in some form. Computing will continue to present challenging career opportunities, and those who work in computing will have a crucial role in shaping the future of society.

It is important for society that the computing disciplines attract quality students from a broad cross-section of people and prepare them to be capable and responsible professionals, scientists, and engineers. Over the years, the professional and scientific computing societies based in the U.S. have taken a leading role in providing guidance and support for higher education in various ways, including the formulation of curriculum standards and guidelines. Several reports that define and update guidelines for computing curriculum have appeared over the past four decades. Recent efforts have targeted international participation, reflecting the need for the leading professional organizations to become truly global in scope and responsibility.

Early in the process that produced *Computing Curricula 2001 (CC2001)*, it became clear that the dramatic expansion of computing that occurred during the 1990s made it no longer reasonable to produce curriculum reports just for computer science and information systems, the two disciplines for which the reports existed previously. Instead, *CC2001* called for a set of reports to cover the growing family of computing-related disciplines, including a volume for each of *computer science, information systems, computer engineering*, and *software engineering*. It was also clear that new computing disciplines would emerge over time and require their own recommendations. Since the publication of *CC2001*, *information technology* has joined the family of computing disciplines and now requires its own curriculum volume.

The *CC2001* report also called for an *Overview Report* to summarize the content of the various discipline-specific reports. This document is the first edition of that *Overview Report*. Its goal is to provide perspective for those in academia who need to understand what the major computing disciplines are and how the respective undergraduate degree programs compare and complement each other. This report summarizes the body of knowledge for undergraduate programs in each of the major computing disciplines, highlights their commonalities and differences, and describes the performance characteristics of graduates from each kind of undergraduate degree program. To create this report, we have examined curriculum guidelines for undergraduate education and have referred to the computing professions and other supporting information as necessary. We have not focused on graduate education or on the identities of the computing research communities. College-level faculty, administrators, and other community leaders are the audience for this report. It outlines the issues and challenges they will face in shaping the undergraduate programs that will serve their constituents and their communities.

In addition, this report includes a *Guide* that offers guidance to a broader audience, including prospective students, their parents and guidance counselors, and others who have reason to care

about the choices that await students who move from high school to college. It provides briefer characterizations of the computing disciplines and profiles factors that prospective students should consider when choosing an area of computing study.

This report is the result of an unprecedented cooperative effort among the leading computer societies and the major computing disciplines. Because things change rapidly in computing, the reports will need frequent updates. Within this report, you will learn how to determine if this is the most recent edition and, if it is not, how to download the newest one.

PART ONE

Chapter 1: Introduction

1.1. Purpose of this report

This report provides an overview of the different kinds of undergraduate degree programs in computing that are currently available and for which curriculum standards are now, or will soon be, available. Teachers, administrators, students, and parents need this report because computing is a broad discipline that crosses the boundaries between science, engineering, and professional practice. In reality, computing consists of several disciplines. Many respected colleges and universities offer undergraduate degree programs in several of *computer science, computer engineering, information systems, information technology, software engineering*, and more. These computing disciplines are related, but are also quite different from each other. The variety of degree programs in computing presents prospective students, educators, administrators, and other community leaders with important choices about where to focus their efforts.

Several questions naturally arise: *What* are *these different kinds of computing degree programs? How are they similar? How do they differ? How can I tell what their names really mean? Which kinds of programs should our local college offer?* And so on. These are all valid questions, but to anyone unfamiliar with the breadth of computing, the responses to these queries may be difficult to articulate**.** This report may help in articulating some answers.

We have created this report to explain the character of the various undergraduate degree programs in computing, and to help you determine which of the programs are most suited to particular goals and circumstances. We intend this report to serve a broad and varied audience. We believe it can be of help to:

- University faculty and administrators who are developing plans and curricula for computing-related programs at their institutions
- Responsible parties in public education, including boards of education, government officials, elected representatives, and others who seek to represent the public interest

In addition, we have included in this report *A Guide to Undergraduate Degree Programs in Computing* (henceforth the *Guide*) with the intent to serve:

- Students who are trying to determine which path of computing study fits their interests and goals
- Parents, guidance counselors, and others who are trying to assist students in their choices
- Professionals who are considering how to continue their education in a rapidly changing, dynamic field
- Anyone who is trying to make sense of the wide range of undergraduate degree programs in computing that are now available.

1.2. Scope of this report

The foundation of this report is the set of curriculum standards that exist for undergraduate degree programs in each of the five major computing-related disciplines mentioned earlier: *computer* *engineering, computer science, information systems, information technology,* and *software engineering*. For each of these disciplines, a curriculum report already exists or will soon exist. Each report represents the best judgment of the relevant professional, scientific, and educational associations, and serves as a definition of what these degree programs should be and do. Those five reports provide the basis for this report. In addition, we have referred to the computing professions and other supporting information as necessary. We have not focused on graduate education or on the identities of the computing research communities.

The remainder of this report includes the following:

- In Chapter 2, we characterize each of the five major computing disciplines.
- In Chapter 3, we flesh out the characteristics of each of these five kinds of degree program and compare them to each other. We also compare and contrast the kind of professional capabilities expected of the graduates of each kind of degree program.
- In Chapter 4, we conclude by alerting educators, administrators, and other responsible parties on some issues that may emerge in the creation of new computing programs.
- In Part Two, we include a *Guide*. This is a short, standalone document that will be published separately from, and distributed more widely than, the whole of this report. In it, we provide information for prospective students, and for those who advise them, to help them make well-informed choices.

Computing itself will continue to evolve. In addition, new computing-related disciplines are likely to emerge. As we update the existing discipline-specific reports and as additional reports for new computing disciplines emerge, you can expect to see updated versions of this report as well. To find out if this document (*CC-Overview)* is the most recent edition of the *Overview Report on Computing Curricula*, go to <http://www.acm.org/education/>. There, you will be able to determine if a newer version exists. If it has, you may download the newest version from that site.

1.3. Background and history

Over the last forty years, four major organizations in the US have developed computing curriculum guidelines for colleges and universities:

- The Association for Computing Machinery (generally called "the ACM" or "the Association for Computing") is a scientific and professional organization founded in 1947. It is concerned with the development and sharing of new knowledge about all aspects of computing (the word "machinery" in its name is just a historical artifact). It has traditionally been the professional home of computer scientists, who devise new ways of using computers and who advance the science and theory that underlies both computation itself and the software that enables it. The ACM began publishing curriculum recommendations in 1968.
- The Association for Information Systems (generally called "AIS"), founded in 1994, is a global organization serving those academics who specialize in Information Systems. AIS is affiliated with Society for Information Management in the U.S., the membership of which consists of IS executives and managers. AIS began providing curriculum recommendations in cooperation with ACM and AITP in 1997.
- The Association for Information Technology Professionals (often referred to as "the AITP") was founded in 1951 as the National Machine Accountants Association. Beginning in 1962, it became the Data Processing Management Association (or DPMA). It adopted its present name in 1996. AITP focuses on the professional side of computing, serving those who use computing technology to meet

the needs of business and other organizations. It began providing curriculum recommendations in 1985.

- The Computer Society of the Institute for Electrical and Electronic Engineers (often referred to as “the IEEE-CS” or “the Computer Society”) became an entity in 1948. As the name suggests, the IEEE (founded originally from American Institute of Electrical Engineers) and the Institute of Radio Engineers in 1884) began as an organization of electrical engineers. The Computer Society is an organization within the IEEE focuses upon on computing from the engineering perspective. Today the Computer Society's members include computer engineers, software engineers, and computer scientists. In recent years, there has been a large overlap in membership between the ACM and the IEEE Computer Society. It began providing curriculum recommendations in 1983.

Prior to the 1990’s, each society produced its own curriculum recommendations. Originally, some separation of effort seemed reasonable, as each society had a special focus:

- The ACM focuses on *the science of computing*, developing both theory and software that stretched the boundaries of what could be done with computers;
- The IEEE-CS focuses on *engineering the computers* to take advantage of new computing knowledge and more advanced software;
- The AITP focuses on *using computers in business*.

Over time, however, the goals of the societies began to overlap, and the advantages of cooperative work among them became obvious. Today, the societies cooperate in creating curriculum standards, and in this way send a single message to their community; many researchers and teachers belong to more than one of the societies.

The ACM and the IEEE-CS joined forces in the late 1980s to create a joint curriculum report for computing. Published in 1991 and known as *Computing Curricula 1991* or CC’91 (CC91), it provided guidelines for curricula for four-year Bachelor’s degree programs in computer science. Throughout the 1990s, various groups made efforts to produce curricula guidelines for other programs in computing education. By 1993, the ACM had produced five reports for two-year Associate degree programs, one report for each of *computer science*, *computer engineering technology*, *information systems*, *computer support services*, and *computing for other disciplines*. [AssocDeg] Also in 1993, the ACM produced curriculum recommendations for a high school curriculum [HS]. In 1997, the ACM, AIS, and AITP [AIS] published a model curriculum and a set of guidelines for four-year Bachelors degree programs in *information systems* [IS97]. The 1990s also saw newer computing disciplines gain increased prominence in the U.S. The discipline called *computer engineering* became more visible in the U.S., as did *software engineering*.

By the end of the 1990s, it was becoming clear that the field of computing had not only grown rapidly but had also grown in many dimensions. The proliferation of different kinds of degree programs in computing left many people confused. Given the growing number of kinds of computing degree programs, confusion was perhaps inevitable. This diversity of computing degrees was a problem that had not existed in a significant way prior to the computing explosion in the 1990s. Because it was new problem, there was no established way of coordinating and simplifying the choices that suddenly seemed to be appearing everywhere.

When the ACM and the IEEE-CS again joined forces in the late 1990s to produce an up-to-date curriculum report to replace CC’91, these organizations could no longer ignore the problem. The original plan called for the two societies to form a joint task force that would update the CC’91 report. ACM and IEEE-CS created a joint task force and its goal was to produce Computing Curricula 2001 (CC2001), a single report that would provide curriculum guidelines for degree programs for the various computing disciplines. However, the members of the task force soon recognized the new reality: computing had grown in so many dimensions that no single view of the discipline seemed adequate. The days when the field of computing consisted of only *computer science* and *information systems* were over, and the richness and breadth provided by the various computing disciplines called for a new way of defining what computing curricula should be.

The CC2001 Task Force faced this challenge by making four important decisions:

1. There should be a curriculum report (or volume) for each of the major computing disciplines, including *computer engineering, computer science, information systems,* and *software engineering;*

2. The number of computing-related disciplines is likely to grow. The curriculum report structure must accommodate not only the four major computing disciplines in existence at that time (enumerated above) but also must accommodate new computing disciplines as they emerge.

3. The growing number of computing disciplines naturally causes confusion in many quarters. Therefore, in addition to the various discipline-specific volumes, there must also be an *Overview* report to serve as a practical "umbrella" guide to the discipline-specific volumes.

4. The pace of change in computing is sufficiently rapid that we must establish a process by which organizations could update curriculum guidelines more frequently than once per decade.

The Task Force recognized that its members were primarily computer scientists and deemed itself qualified to produce a report only for computer science. It called for the ACM, the IEEE-CS, the AIS, and other professional societies to cooperate in efforts to create volumes for computer engineering, information systems, and software engineering. The work of this task force, known as *Computing Curricula 2001* (CC2001), was published in December 2001 [CC2001]. The CC2001 Report contains two specifications:

- Provide a new structure for computing curriculum guidelines encompassing the decisions taken by the Task Force and listed above in the CC2001 model.
- Detailed curricula guidelines for undergraduate degree programs in computer science.

[Because the *CC2001* report included CS curriculum guidelines, those who refer to it for its computer science content might think of as *CS2001.* Beginning with the publication of this report, we will use the title "Computing Curricula 20xx" for Overview reports. "Computer Science 20xx" will designate new editions of CS curriculum guidelines. In all cases, "20xx" will be the year of publication.]

In response to the *CC2001* model, work soon began on other discipline-specific volumes:

- The *information systems* community published its updated report in 2002. [IS-2002]
- The *software engineering* group has completed its report and it expects publication in 2004. We thus refer to it as SE-2004.
- The *computer engineering* report is nearing completion. We expect its publication in early 2005 and thus refer to it as CE-2004.
- The CC2001 prediction of additional emerging computing disciplines has already proved correct. A report on degree programs in *information technology* is under development. We anticipate that it will be published in 2005 and thus refer to it as IT-2005.

The diagram in *Figure 1.1* represents the scope of the continuing effort to provide guidelines and standards for computing curricula. The top-level *Overview* block, CC2004, represents this report. Each of the first five sub-blocks represents a curriculum report for one of the existing computing disciplines. The sixth sub-block is a placeholder for future reports on additional computing disciplines as necessitated by the emergence of new computing disciplines.

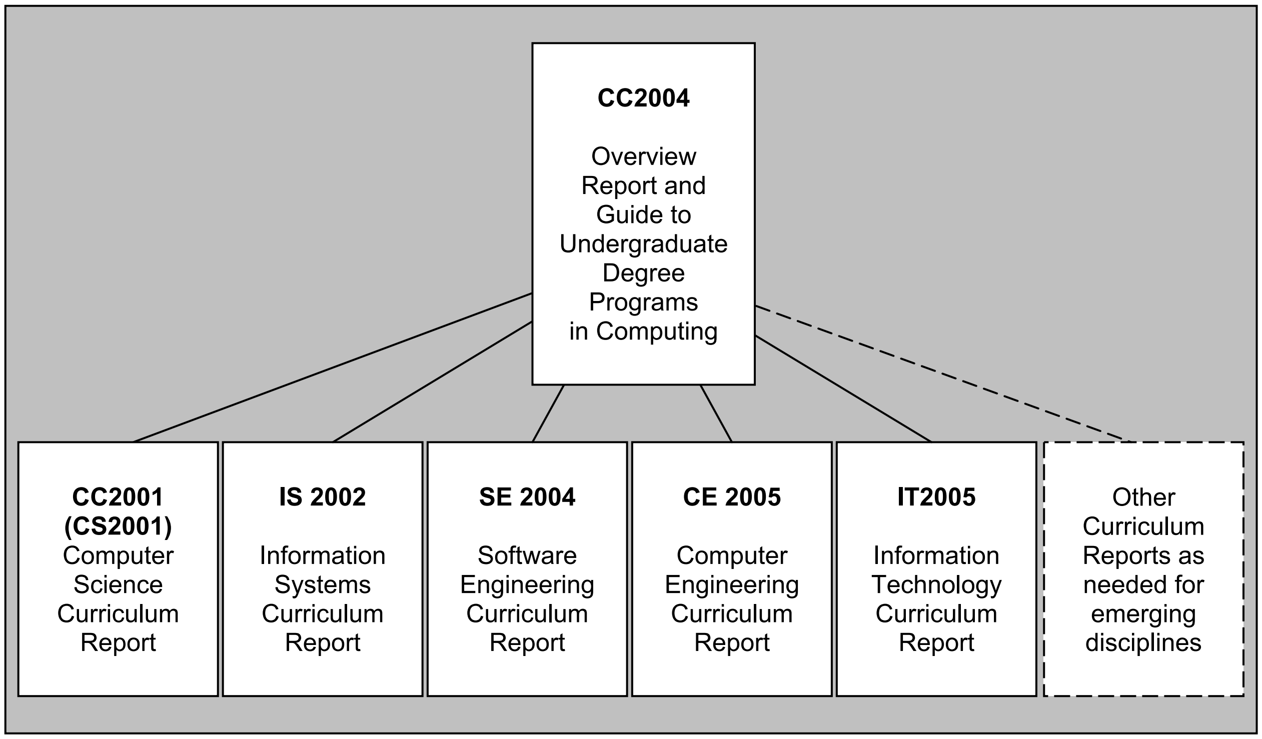
CC2004
Overview Report and Guide to Undergraduate Degree Programs in Computing
CC2001 (CS2001) Computer Science Curriculum Report
IS 2002 Information Systems Curriculum Report
SE 2004 Software Engineering Curriculum Report
CE 2005 Computer Engineering Curriculum Report
IT2005 Information Technology Curriculum Report
Other Curriculum Reports as needed for emerging disciplines

Figure 1.1. Structure of the Computing Curriculum Volumes

1.4. Guiding Principles

Five principles guided the development of this report.

1. ***The dramatic growth in the number of computing disciplines, and in their collective impact on society, requires that the computing disciplines articulate a shared identity.*** Given the importance of computing to society, we in computing have a responsibility to help society understand what we do. The fact that computing offers several kinds of academic programs is a major strength and an opportunity but requires that we offer society a practical vision of our shared field, of the various disciplines within it, and of the meaningful choices that face students, educators, and their communities. The goal of this report is to articulate the shared identity, the separate identities of each computing discipline, and the choices available to students, educators, and communities.

2. ***Each computing discipline must be a participant in defining the identities and choices as articulated in this report.*** Each computing discipline must articulate its own identity, recognize the identities of the other disciplines, and contribute to the shared identity of computing.

3. ***This report must address a broad audience, not just its technically oriented constituents***. As discussed in Section 1.1, the audience for this report includes a range of people who have reason to become familiar with academic computing degree programs. Most members of that audience are not computing educators. Our goal is to paint a concise and useful picture that will illuminate the choices faced by students and by those who are responsible for shaping their educational choices. This goal is fundamentally different from the goal of reports that define curriculum guidelines for degree programs. It dictates that we must be relatively concise and that we minimize technical jargon. We ask the technically oriented reader to appreciate our need to avoid the kind of distinctions and technical emphasis expected of documents aimed at a technical audience.

4. ***We should characterize the computing disciplines by reference to the body of knowledge and skills defined in the most recent curriculum report for each of those disciplines.*** The definition of a shared characterization of the computing disciplines is unprecedented, and it is imperative that we set attainable goals. We confine our attention to the bodies of knowledge and skills defined by each computing discipline as published in the individual curriculum reports; we do not consider pedagogy or course definition. We believe that pedagogical issues and the definition of computing courses that might serve multiple audiences across the computing disciplines are important and timely concerns. However, we believe it would be ill advised to address such issues in this report. [Note: We should not construe this decision as a precedent for others to follow. It is possible that authors of subsequent reports want to revisit this issue.]

5. ***This report must go beyond an examination of details to generate a useful synthesis for the intended audience.*** While the findings of this report are based on examination of the bodies of knowledge in current discipline-specific curriculum volumes, we must go beyond simple examination-and-reporting to generate a synthesis that will be meaningful and useful for our audience. Our task requires representatives of each discipline to make judgments about how to form an insightful, consensus-based overview of the computing disciplines.

Chapter 2. Computing Disciplines

We now focus our attention on computing disciplines. There may be dozens if not hundreds around the world. However, among them, five appear to have some prominence today. These include computer engineering, computer science, information systems, information technology, and software engineering. We will show how these disciplines evolved, their commonalities and differences, and their scope within the computing community.

2.1. What is computing?

In a general way, we can define computing to mean any activity of a technical nature involving computers. Thus computing includes hardware, software, and communications that involve the design as well as the use of these. It includes the design and building of hardware and software systems for a variety of purposes and it includes the management and structuring of a whole range of information perhaps in different formats (e.g. text, video, sound, etc). Computing also it includes the processing of information, the protection and the care of that information and it includes the usability of computer systems, making them behave intelligently (however that is to be interpreted) and so on. The possibilities are just vast. Computing also has other meanings that are more specific, based on the context in which the term is used. For example, an information systems specialist will view computing somewhat differently from a software engineer. Hence, we can describe computing as a discipline associated with the structuring and the organization of information as well as the automatic processing of that information.

Computing provides a wide range of choices about how an individual might focus his or her professional life. To prepare for entry into a computing profession, a student typically earns a bachelors degree in one of the computing disciplines. There are currently five major kinds of undergraduate degree programs in computing, and each one provides a different perspective on the broad topic area. In the next section, we shall see what these five computing disciplines are, and how they compare with each other in terms of their focus as well as the kinds of problems and issues they address.

2.2. The landscape of the computing disciplines

Computing is not just a single discipline but is a family of disciplines. During the 1990s, important changes in computing and communications technology, and in the impact of that technology on society, lead to important changes in this family of disciplines.

2.2.1. Before the 1990s

Before the 1990s, only three computing-related disciplines were highly visible in North America: computer science, electrical engineering, and information systems. Each of these disciplines was concerned with its own well-defined area of computing. Because they were the only prominent computing disciplines, and because each one had its own area of work and influence, it was much easier for students to tell which kind of degree program to choose. For students who wanted to become expert in developing software or with the theoretical aspects of computing, computer science was the obvious choice. For students who wanted to work with hardware, electrical engineering was the clear option. For students who wanted to use hardware and software to solve business problems, information systems was the place to be.

Each of these three disciplines had its own domain. There was not any shared sense that they constituted a family of computing disciplines. As a practical matter, computer scientists and electrical engineers sometimes worked closely together, as they were both concerned with developing new technology. The often shared facilities and sometimes required the help of each other. Information systems specialists had ties with business schools and did not have much interaction with computer scientists and electrical engineers. The pre-1990s world of the computing disciplines appears in Figure 2.1, with the distance between the disciplines indicating how closely they worked with each other.

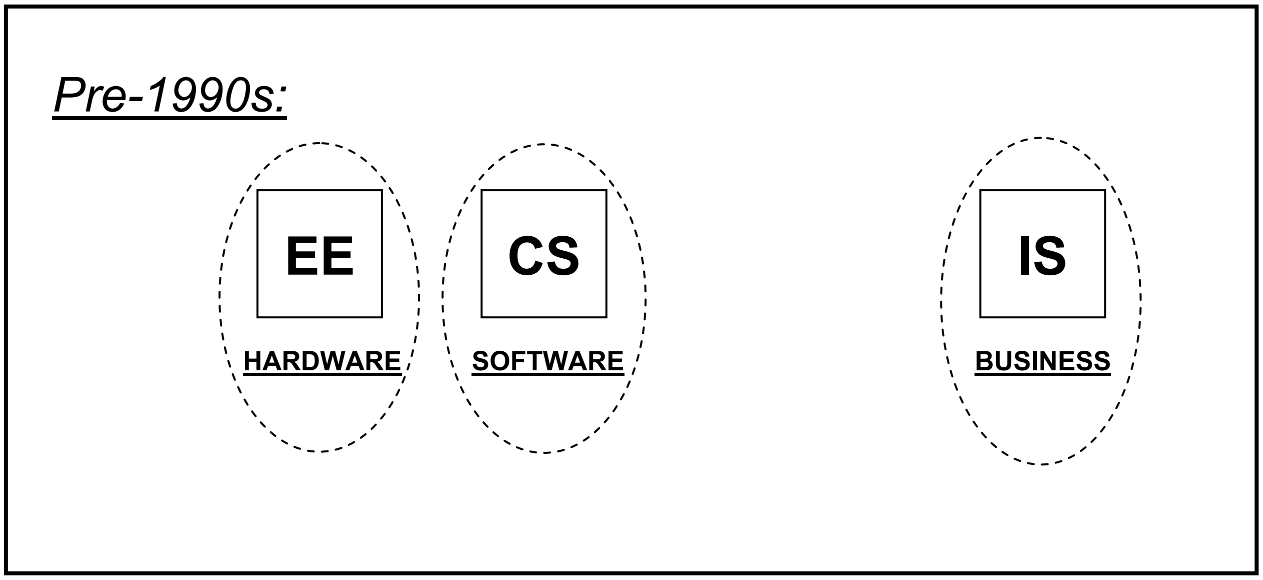

Figure 2.1: The landscape of mainstream computing degree programs, pre-1990.

2.2.2. Significant developments of the 1990s

During the 1990s, several developments changed the landscape of the computing disciplines in North America, although in other parts of the world some of these changes occurred earlier:

- *Computer engineering* became a strong discipline. While computer engineering had a significant presence in some universities prior to the 1990s, at most universities with engineering programs it was one of several specialty areas within electrical engineering. During the 1990s, computer chips became basic components of most kinds of electrical devices and many kinds of mechanical devices. (For example, modern automobiles contain several computers that perform tasks that are transparent to the driver.) Computer engineers design and program the chips that permit digital control of many kinds of devices. The dramatic expansion in the kinds of devices that rely on chip-based digital logic caused computer engineering to become a strong field unto itself, with degree programs at many US universities. In other countries titles like *computer systems engineering* were often used instead.

- *Software engineering* emerged as an area within *computer science*. As computing is used to attack a wider range of complex problems, creating reliable software becomes more difficult. With large, complex programs, no one person can understand the entire program, and various parts of the program can interact in unpredictable ways. (For example, fixing a bug in one part of a program can create new bugs elsewhere.) People also use computing in safety-critical tasks, where a single bug can cause injury or death. Over time, it became clear that producing good software is very difficult, very expensive, and very necessary. This lead to the creation of software engineering, a term that emanated from a NATO sponsored conference held in Garmish, Germany in 1968. While computer science (like other sciences) focuses on creating new knowledge, software engineering (like other engineering disciplines) focuses on rigorous methods for designing and building things that reliably do what they're supposed to do. Major conferences on software engineering were held in the 1970s, and during the 1980s, many computer science degree programs included software engineering courses. However, it was not until the 1990s that one could reasonably expect to find software engineering as a key component of computer science study at nearly every institution.

- *Software engineering* began to develop as a discipline unto itself. Originally the term 'software engineering' was introduced to reflect the utilization of traditional ideas from engineering to the problems of building software. As software engineering matured, the scope of its challenge became clearer. In addition to its computer science foundations, software engineering also involves human processes that, by their nature, are less abstract and harder to formalize than are the logical abstractions of computer science. Experience with software engineering courses within computer science curricula showed many that such courses can teach students "about the field of software engineering" but do not succeed at teaching them "how to be software engineers". Many experts concluded that the latter goal requires a range of coursework and applied project experience that go beyond what they could add to the computer science curricula. Degree programs in software engineering emerged in the United Kingdom and Australia during the 1980s, but these programs were in the vanguard. In the United States, degree programs in software engineering, designed to provide a more thorough foundation than can be provided within computer science curricula, began to emerge during the 1990s.

- *Information systems* had to address a growing sphere of challenges. Prior to the 1990s, many information systems specialists focused primarily on the computing needs that the business world had faced since the 1960s: accounting systems, payroll systems, inventory systems, etc. By the end of the 1990's, networked personal computers had become basic commodities. No longer tools only for technical specialists, they became integral parts of the work environment, used by people at all levels of the organization. Because of the expanded role of computers, organizations had more information available than ever before and organizational processes were increasingly enabled by computing technology. The problems of managing information became extremely complex, and the challenges of making proper use of information and technology to support organizational efficiency and effectiveness became crucial issues. Because of these factors, the challenges faced by information systems specialists grew in size, complexity, and importance. In addition, Information Systems as a field paid increasing attention to the use of computing technology as a means for communication and collaborative decision making in organizations.

- *Information technology* programs began to emerge. During the 1990s, computers became essential work tools at every level of most organizations and networked computer systems became the informational backbone of organizations. We credit this to improving productivity. However, it also created new workplace dependencies, as problems in the computing infrastructure can limit employees' ability to do their work. IT departments emerged to ensure that the computing infrastructure of an organization was suitable, that it worked reliably, that people in the organization had their computing-related needs met and had their problems fixed. As it became clear that academic degree programs were not producing graduates who had the right mix of knowledge and skills to meet these essential needs, many colleges and universities developed degree programs in information technology to fill this crucial void.

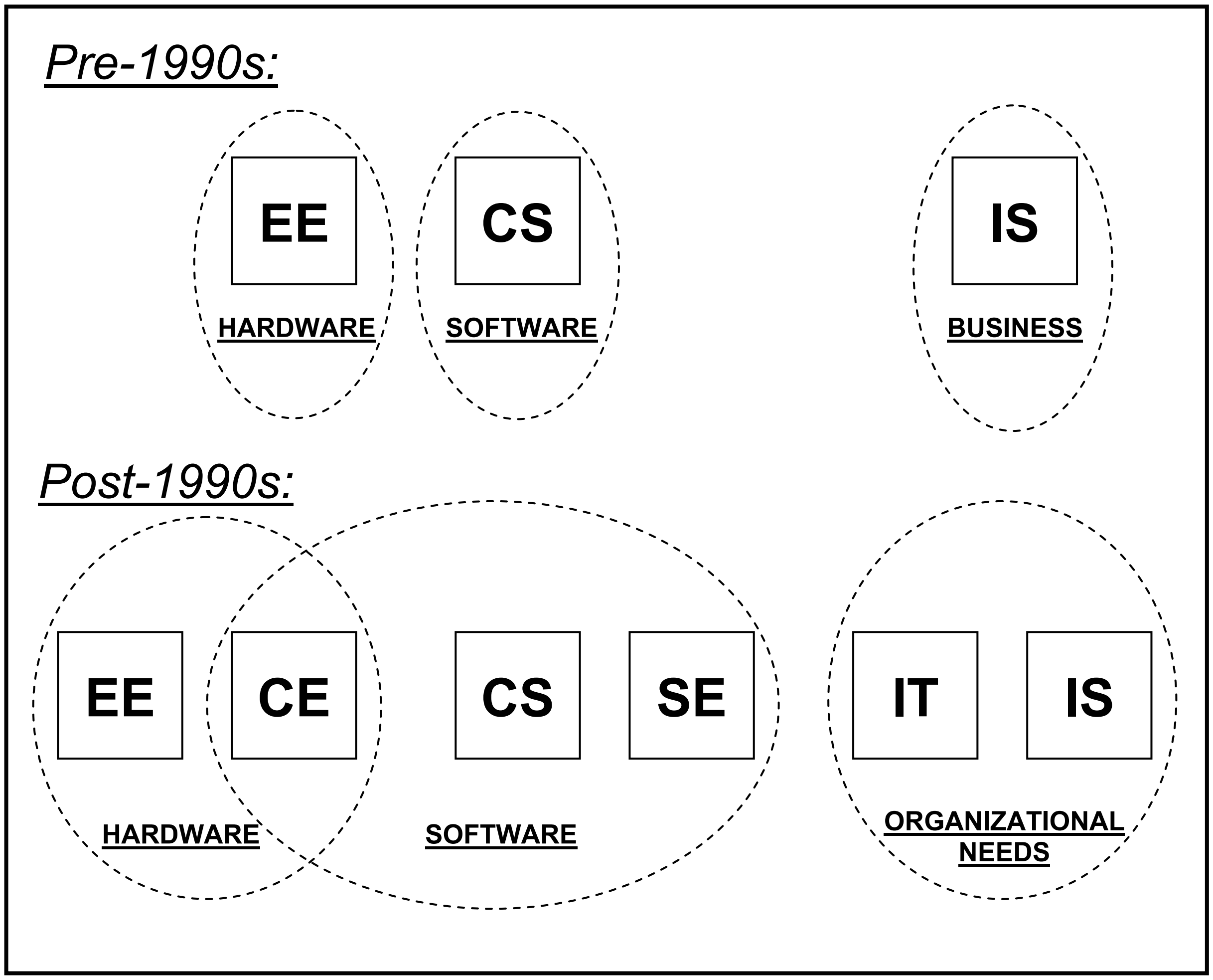

Figure 2.2. The computing disciplines, before and after the 1990s.

Collectively these developments reshaped the landscape of the computing disciplines. The explosive growth of the internet affected strongly all computing disciplines. Furthermore, the tremendous resources that were allocated to information technology projects in all Western societies because of various factors, including the dot.com bubble, the anticipated Y2K problems, and in Europe the launch of the Euro.

2.2.3. After the 1990s

The new landscape of the computing disciplines reflects the ways in which computing-as-a-whole has matured to address the important problems of the new millennium. *Computer engineering* emerged from *electrical engineering* and assumed a primary role with respect to computer hardware and related software. *Software engineering* emerged from within *computer science* to address the important challenges inherent in building complex software systems that are reliable and affordable. *Information technology* came out of nowhere to fill a void that other computing disciplines did not adequately address.

While this maturation is a positive evolution, there is a greater range of possibilities for students and educational institutions to focus their attention. The increased diversity of computing programs means that the spheres of responsibility are somewhat different from what they were before the 1990s. Figure 2.2 shows the pre-1990s and post-1990s spheres of responsibility. As discussed in section 2.2.1, before the 1990s a convenient one-to-one mapping allowed students to determine which discipline matched their personal interests. This neat mapping is evident in the top portion of Figure 2.2. As shown in the bottom portion of Figure 2.2, the post-1990s world of computing presents greater possibilities and opportunities.

As a practical matter, it is still clear where students who want to study hardware should go. *Computer engineering* has emerged from *electrical engineering* as the home for those working on the hardware and software issues involved in the design of digital devices. For those with other interests, however, the choices are not so clear-cut. In the pre-1990s world, students who wanted to become expert at software development would go into computer science. The post-1990s world presents meaningful choices: *computer science, software engineering*, and even *computer engineering* each include their own discipline-specific perspective on software development. Similarly, in the pre-1990s world, the predominant area for applying technology to real-world problems was on business and *information systems* was the home for such work. The scope of real-world applications has broadened from business to organizations of every kind, and students face a choice; for instance, information *systems* and *information technology* programs would each have its own special focus.

2.3. Descriptions of the major computing disciplines

In this section, we characterize each of the five major kinds of computing disciplines. See sections 2.6 and 2.7 for more information on how to understand this important distinction between the *names of the computing disciplines* and the *names of a particular degree program*.

2.3.1. Computer Engineering

Computer engineering is concerned with the design and construction of computers, and computer based systems. It involves the study of hardware, software, communications, and the interaction between them. Its curriculum focuses on the theories, principles, and practices of relevant areas of traditional electrical engineering and mathematics, and applies them to the problems of designing computers and the many kinds of computer-based devices.

Computer engineering students study the design of digital hardware systems, including computers, communications systems, and devices that contain computers. They also study software development with a focus on the software used within and between digital devices (not the software programs directly used by computer users). The emphasis of the curriculum is on hardware more than software, and it has a very strong engineering flavor.

Currently, a dominant area within computing engineering is embedded systems, the development of devices that have software components embedded in hardware. For example, devices such as cell phones, digital recorders, alarm systems, x-ray machines, and laser surgical tools all require integration of hardware and embedded software, and they are all the result of computer engineering.

2.3.2. Computer Science

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 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 lets them develop new problem-solving approaches that provide better performance.

- They devise new ways to use computers. Progress in the CS areas of networking, database, and human-computer-interface came together as the world-wide-web, which changed the world. Now, researchers are working to make robots be practical aides and even demonstrate intelligence, databases create new knowledge and, in general, use computers to do new things.
- They design and implement software. Computer scientists take on challenging programming jobs. They also supervise other programmers, keeping them aware of new approaches.

Computer science spans the range from theory to programming. Other disciplines can produce graduates better prepared for specific jobs, while computer science offers a comprehensive foundation that permits graduates to adapt to new technologies and new ideas.

2.3.3. Information Systems

Information systems specialists focus on integrating information technology solutions and business processes to meet the information needs of businesses and other organizations and enable organizations to achieve their objectives in an effective and efficient way. This discipline's perspective on "Information Technology" emphasizes *information*, and sees *technology* as an instrument to enable the generation, processing and distribution of needed information. Professionals in this discipline are primarily concerned with the information that computer systems can provide to aid the organization in defining and achieving its goals and the processes that organizations can implement using information technology. Information systems professionals often work in organizations that are large and complex, and with information systems that are correspondingly large and complex. They understand both technical and organizational factors, and must be able to help the organization determine how information and technology-enabled business processes can provide the organization with a competitive advantage.

The discipline now called *information systems* began more than forty years ago to address the data processing needs of business in the areas of accounting, payroll, and inventory. As the role of computing has expanded throughout the organization, so has the scope of information systems. Today, the information systems specialist plays a key role in determining the requirements for an organization's information systems and is active in their specification, design, and implementation. As a result, such professionals require a sound understanding of organizational principles and practices so that they can serve as an effective bridge between the technical and management communities within an organization, enabling them to work in harmony to ensure that the organization has the information and the systems it needs to support its operations. Information systems professionals are also involved in designing technology-based organizational communication and collaboration systems.

Most departments offering programs in *Information Systems* (IS) are located in business schools, and most IS degrees are combined computing and business degrees. A wide variety of IS programs exists under various labels which often reflect the nature of the program. For example, programs in Computer Information Systems usually have the strongest technology focus, whereas programs in Management Information Systems sometimes emphasize organizational and behavioral aspects of the IS discipline. The names of the degree programs are not consistent. Therefore, it is important to evaluate the details of the curriculum that a specific program follows to understand how its purpose.

2.3.4. Information Technology

Information technology is a label that has two meanings. In the broadest sense, we often use "information technology" interchangeably with "computer technology". In a more focused sense, it refers to academic degree programs that prepare students to meet the technology needs of business, government, healthcare, schools, and other kinds of organizations.

In the previous section, we said that the field of *Information Systems* focuses on the "information" aspects of "information technology". The field of *Information Technology* is the complement of that perspective. IT's emphasis is on the technology itself more than on the information it conveys. IT is a new and rapidly growing discipline, which started as a grass roots response to the practical, everyday needs of business and other organizations. Today, organizations of every kind are dependent on information technology. They need to have the appropriate systems in place. Those systems must work properly and be secure. Professionals must upgrade, maintain, and replace them as appropriate. The people who work throughout an organization require support from IT staff that thoroughly understands computer systems and are committed to solving whatever computer-related problems they might have. Graduates of information technology programs address these needs.

Degree programs in Information Technology arose because degree programs in the other computing disciplines failed to produce an adequate supply of graduates capable of handling these very real needs. IT programs exist to produce graduates who possess the right combination of knowledge and practical, hands-on expertise to take care of both an organization's information technology and the people who use it. IT specialists assume responsibility for selecting hardware and software products appropriate for an organization, integrating those products with organizational needs and infrastructure, and installing, customizing and maintaining those applications for the organization's computer users. Examples of these responsibilities include the installation of networks; network administration and security; the design of web pages; the development of multimedia resources; the installation of communication components; the oversight of email products; and the planning and management of the technology life-cycle by which an organization's technology is maintained, upgraded, and replaced.

2.3.5. Software Engineering

Software engineering is the discipline of developing and maintaining software systems that behave reliably and efficiently, and are affordable to develop and maintain. This reflects its origins as outlined in section 2.2.2. However, more recently it has evolved in response to the increased importance of software in safety-critical applications and to the growing impact of large and expensive software systems in a wide range of situations.

Traditionally, computer scientists produced software, and electrical engineers produced the hardware on which the software runs. As the size, complexity, and critical importance of software grew, so did the need to ensure that software performs as intended. By the early 1970s, it was apparent that proper software development practices require more than just the underlying principles of computer science; they also need the rigor that the engineering disciplines bring to the reliability and trustworthiness of the artifacts they engineer.

Software engineering is different in character from other engineering disciplines, due to both the intangible nature of software and to the discontinuous nature of software operation. It seeks to integrate the science of computer science with the engineering principles developed for tangible, physical phenomena. Prospective students can expect to see software engineering presented in two contexts:

- Degree programs in computer science offer one or more software engineering courses as elements of the CS curriculum. In addition, most programs offer a multi-course concentration in software engineering within the computer science discipline.
- A number of institutions offer a software engineering degree program.

Degree programs in computer science and in software engineering generally have many courses in common. Software engineering students generally study more applied mathematics and less theory than computer science students do. They tend to take a more rigorous and pragmatic view of software reliability and maintenance. While computer science students might study areas such as artificial intelligence or computer graphics, software engineering students focus more on techniques for developing and maintaining software that is correct from its inception to avoid costly and potentially dangerous situations later. While CS students are likely to have heard of the importance of such techniques, the engineering knowledge and experience provided in SE programs goes beyond what CS programs can provide. Such is the importance of this that one of the recommendations of the SE report is that during their program of study students of SE should participate in the development of software to be used in earnest by others from some significant application domain. Thus knowing how to provide genuinely useful and usable software is of paramount importance.

In the workplace, "software engineer" is a job label. There is no standard definition for this term when used in a job description. The role of a "software engineer" varies widely among employers. It can be a title equivalent to "computer programmer" or a title for someone who manages a large, complex, and/or safety-critical software project. The public must be mindful not confuse the discipline of software engineering with the ambiguous use of the term 'software engineer" as used in employment advertisements. The two terms are quite often very different meanings.

2.4. Conceptual snapshots: A graphical view of the computing disciplines

To help you understand the commonalities and differences among the computing disciplines, we created a set of graphical characterizations of them. These graphics are just snapshots to give you a better feel for how the disciplines compare to each other. They provide a simple view of how the various disciplines occupy the "problem space" of computing. To keep things simple, we have used the diagram shown in Figure 2.4.

The horizontal axis ranges from "Theory, Principles, Innovation" on the left, to "Application, Deployment, Configuration" to the right. Thus, someone who likes the idea of working in a laboratory to invent new things, or in a university to develop new principles will want to work in a discipline that occupies the space to the left. Conversely, someone who wants to work with people to help them choose and use appropriate technology, or who wants to learn how to integrate off-the-shelf products to solve key organizational problems, will want an area that occupies space to the right. Because there are many, many kinds of jobs and tasks that fall between these two extremes, one should not just look only at the far left and far right, but rather consider the range of possibilities in between those extremes.

The vertical axis ranges from "Computer Hardware and Architecture" at the bottom, to "Organizational System Issues" at the top. As we move higher on this axis, the focus of work is more on people and on information that means something to people. As we move lower on this axis, the focus of work is more on devices and on the data shared among them. Thus, someone who likes designing and building circuits, or who is fascinated with the inner workings of computers, will care about the lower portion of the space, while someone who likes seeing how technology can work for people, or who is curious about the impact of technology and information on organizations, will care about the upper portion of the space.

We can consider both the horizontal and vertical dimensions at once. For example, someone who cares about making things work for people, but is more interested in devices than information or organizations will be interested in the lower-right, someone who wants to develop new theories about how information affects organizations will be interested in the upper-left, and so on. In Figures 2.5 through 2.9, we use this framework to sketch out the conceptual territory occupied by each of the five computing disciplines.

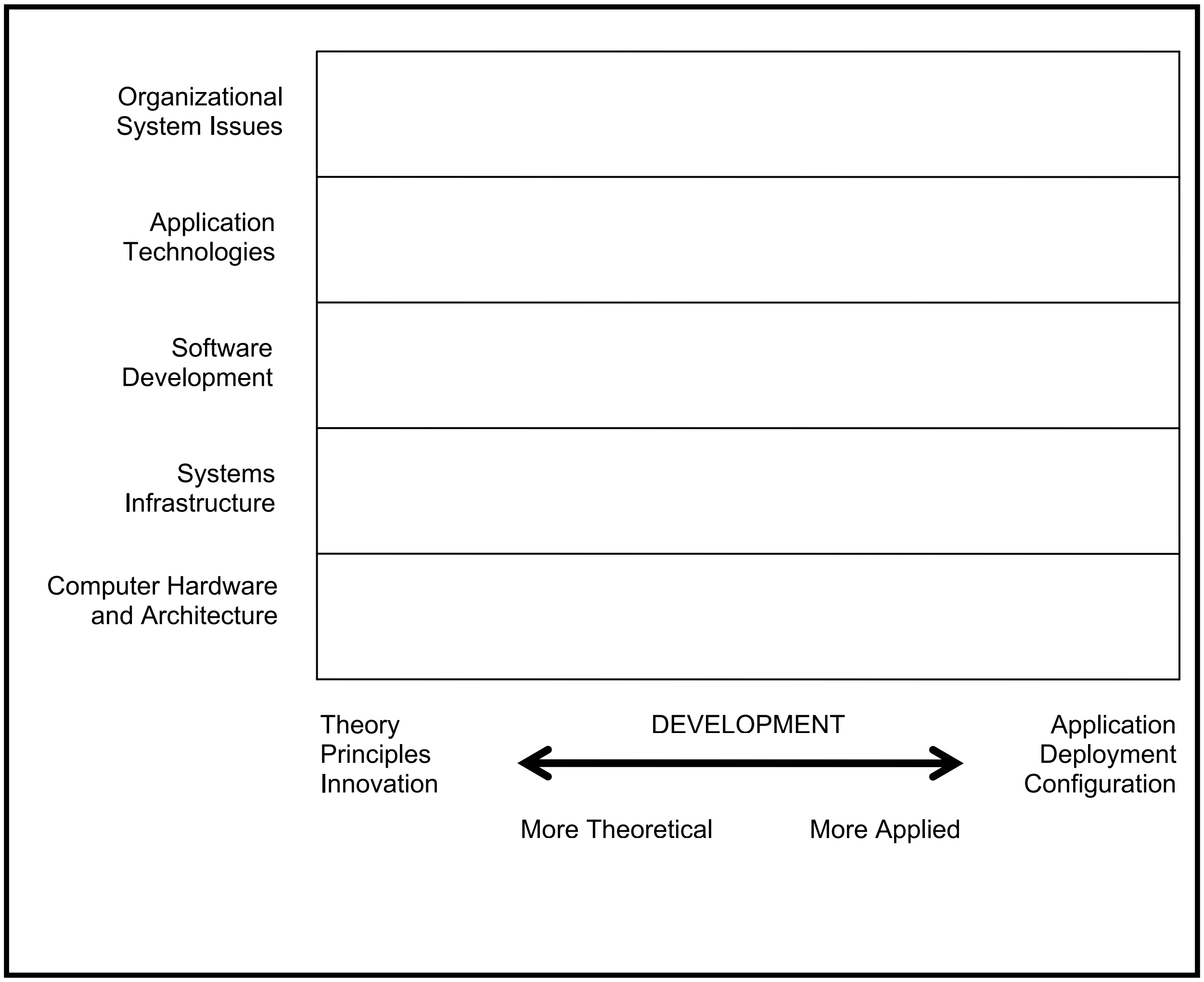
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied

Figure 2.4. The problem space of computing

2.4.1. Computer Engineering

The shaded portion in Figure 2.5 represents the *computer engineering* discipline. It is broad across the bottom because computer engineers cover the range from theory and principles to practical application with respect to designing and implementing hardware solutions and the software that lives inside the hardware. It narrows towards the center as we move upwards because computer engineers' interests narrow as we move away from the hardware. By the time we get up to the level of software development, we see that computer engineers tend not to be developing new principles about software, nor are they writing software programs that everyday people will use. Rather, their interest tends to be narrowed to horizontal center, because they care about software only inasmuch as they need it to develop integrated devices.

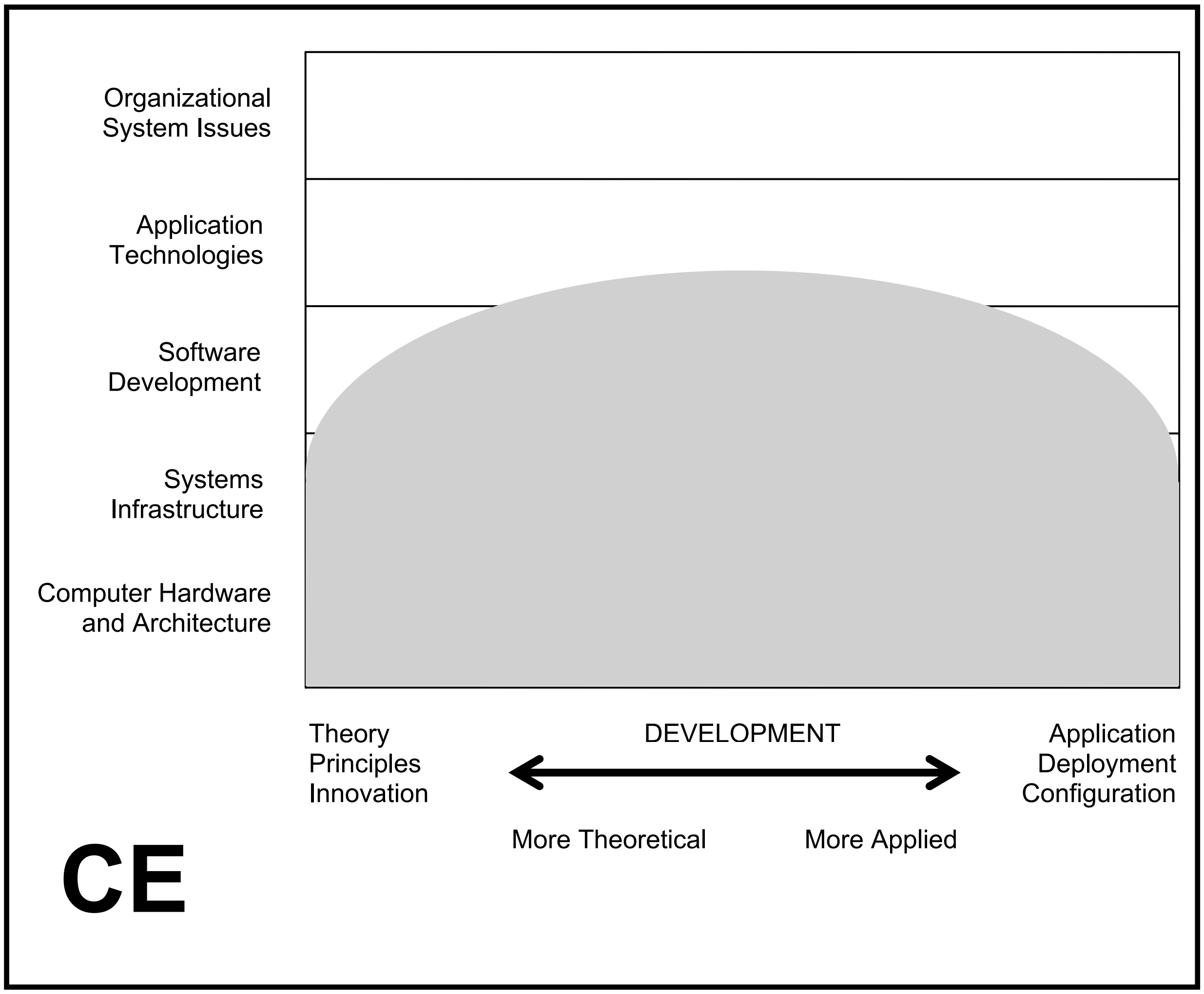
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
CE

Figure 2.5. Computer Engineering

2.4.2. Computer Science

The shaded portion in Figure 2.6 represents *computer science*. Computer science covers most of the vertical space between the extreme top and extreme bottom because computer scientists generally do not deal with "just the hardware" that runs software, or about "just the organization" that make use of the information that computing can provide. As a group, computer scientists care about almost everything in between those areas (down as far as the software that enables devices to work; up as far as the information systems that help organizations operate). They design and develop all types of software, from systems infrastructure (operating systems, communications programs, etc.) to application technologies (web browsers, databases, search engines, etc.) Computer scientists create these capabilities, but they do not manage the deployment of them. Therefore, the shaded area for computer science narrows and it then stops as we move to the right. This is because computer scientists do not help people to select computing products, nor to tailor products to organizational needs, nor do they learn to use such products.

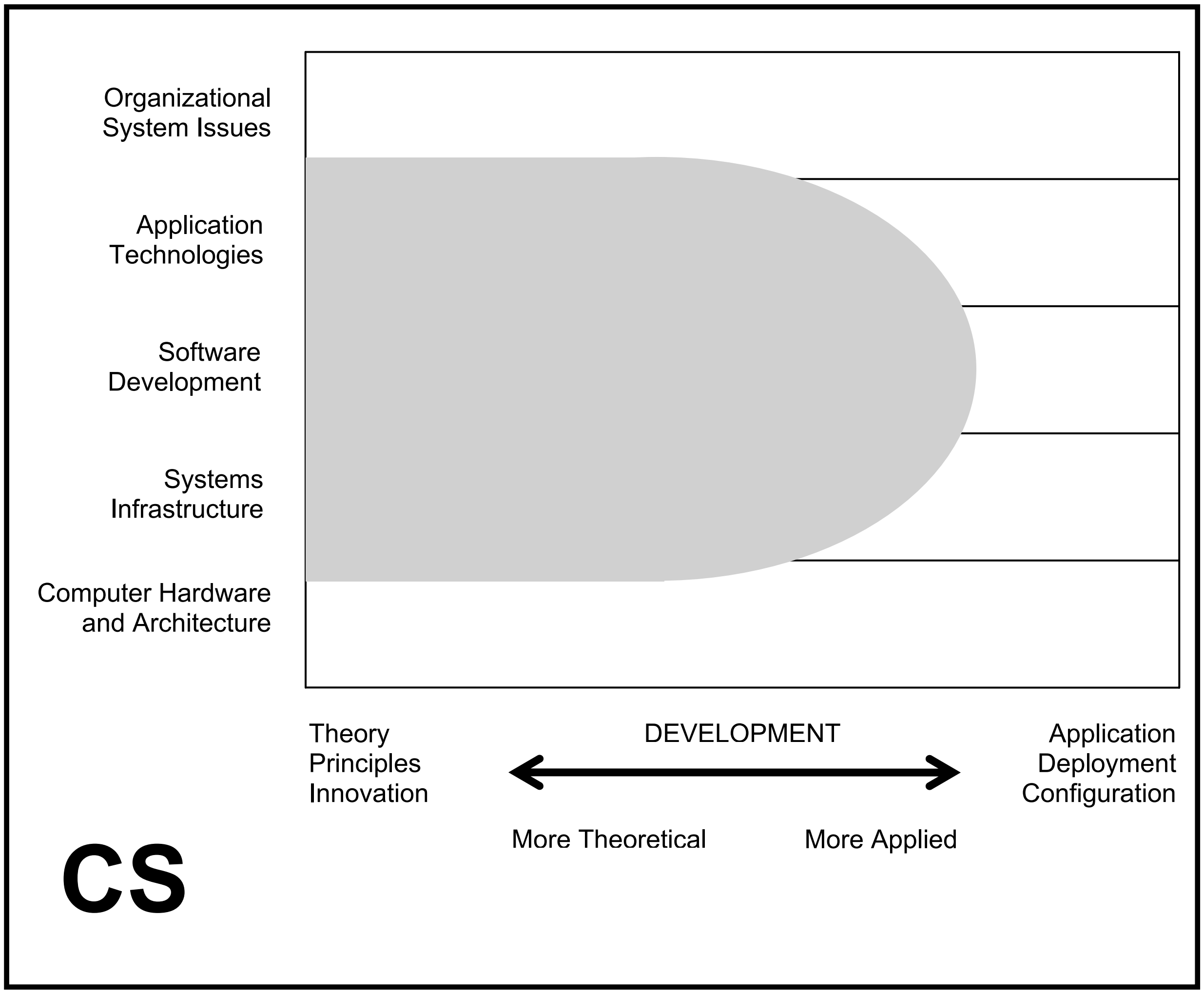
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
CS

Figure 2.6. Computer Science

2.4.3. Information Systems

The shaded portion in Figure 2.7 represents the discipline of *information systems*. The shaded area extends across most of the top-most level because IS people are concerned with the relationship between information systems and the organizations that they serve in a very broad way, extending from theory and principles to application and development; many IS professionals are also involved in system deployment and configuration and training users. The area covered by IS dips downward, all the way through software development and systems infrastructure in the right half of the graph. This is because IS specialists often tailor application technologies (especially databases) to the needs of their organizations, and they often write software that combines other software in ways that suit their organizations' needs for information. For IS specialists, the primary customer is the organization itself, and their job is to ensure the organization has the information it needs and the most effective and efficient organizational processes supported by the most suitable technology in place.

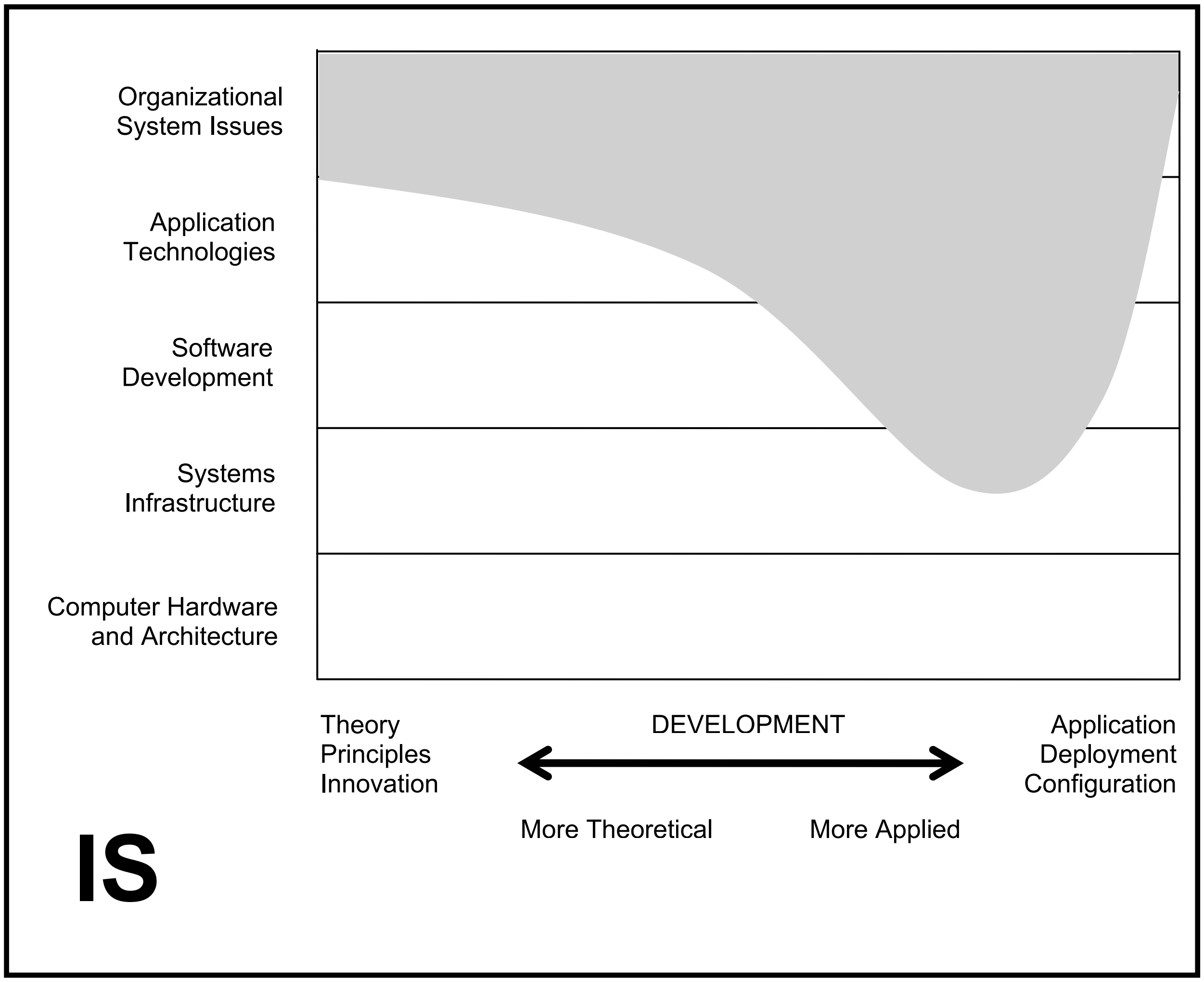
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
IS

Figure 2.7. Information Systems

2.4.4. Information Technology

The shaded portion in Figure 2.8 represents the *information technology* discipline. Its shaded area extends down most of the right edge, as it focuses on the application, deployment, and configuration needs of organizations and people over a wide spectrum. Across this range (from organizational information systems, to application technologies, and down to systems infrastructure), their role has some overlap with IS, but IT people are unique in their often have a focus on satisfying human needs that arise from computing technology. In addition, IT's shaded area goes leftwards, from application into development, especially in the area of application technologies. This is because IT people often develop the web-enabled digital technologies that organizations use for a broad mix of informational purposes.

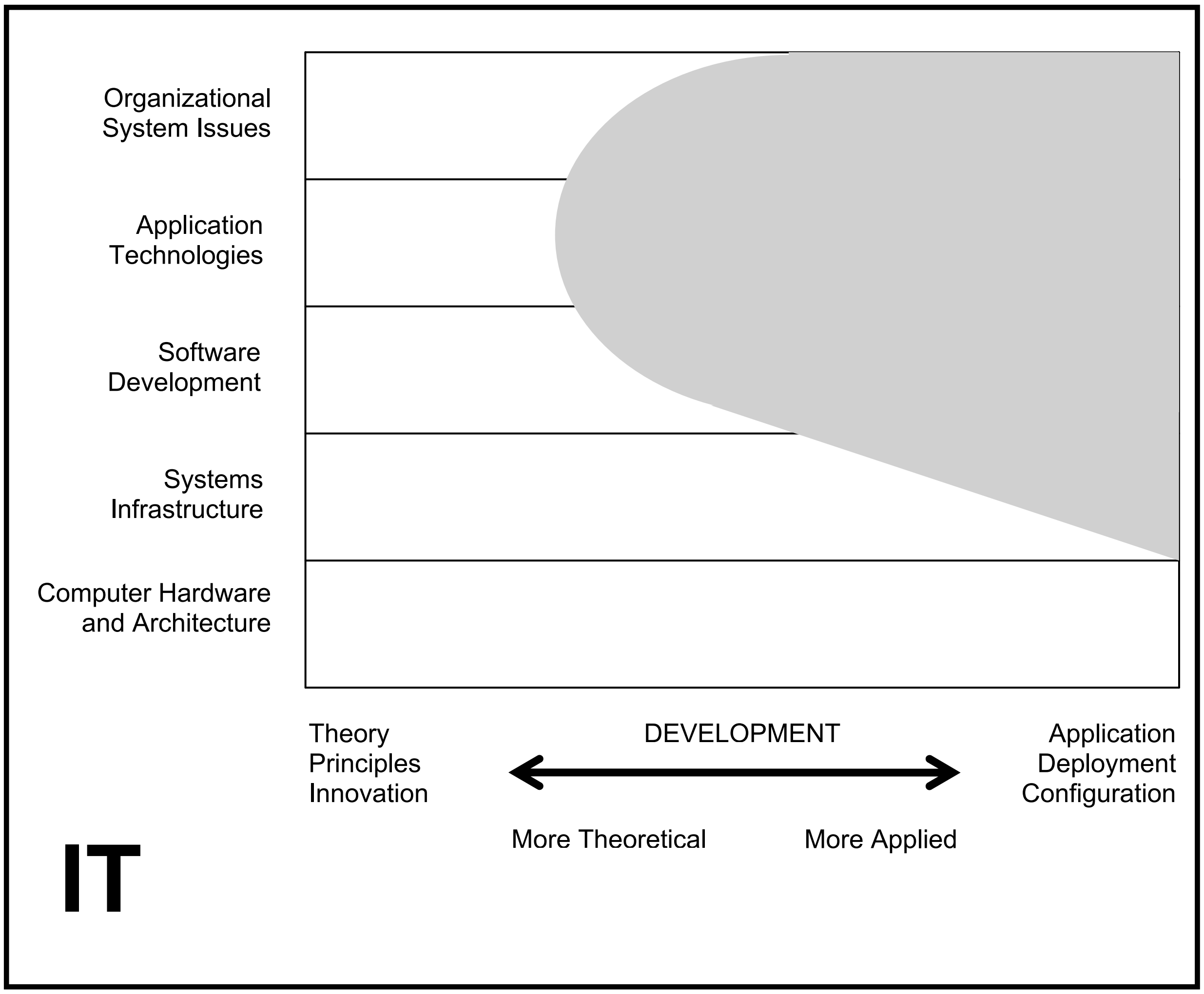
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
IT

Figure 2.8. Information Technology

2.4.5. Software Engineering

The shaded portion in Figure 2.9 represents the discipline of software engineering. Just as we have seen computer engineering's area span the entire horizontal dimension at the lower hardware-related level, and IS span most of that dimension at the higher organization-related level, software engineering covers a wide range with respect to the systematic development of software. This is because SE people have stepped up to fill a wide range of needs in large-project software expertise. SE's main goal is to develop systematic models and reliable techniques for producing high-quality software, and these concerns extend all the way from theory and principles to daily practice. The domain of SE also extends downward through systems infrastructure, as SE people develop software infrastructure that is robust in operation. Its domain also extend upward into information systems because SE people are interested in designing and developing information systems that help them manage software development projects to ensure that correct software is produced on time and within budget.

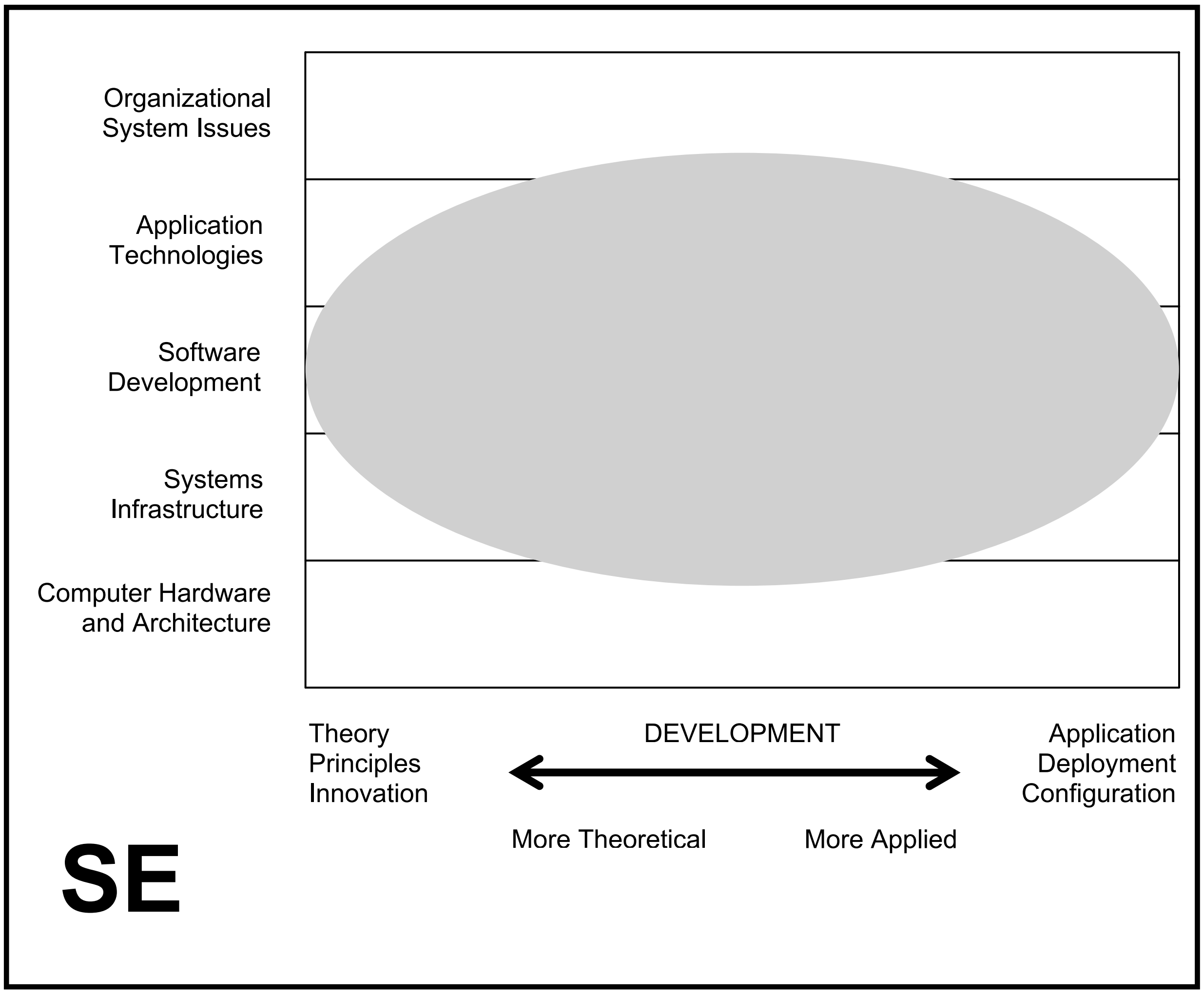
Organizational System Issues
Application Technologies
Software Development
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
SE

Figure 2.9. Software Engineering

Chapter 3: Degree programs and career requirements

In this chapter, we summarize the characteristics of degree programs in each of the five major disciplines and compare them to each other in terms of both (a) the relative focus of coverage within degree programs, and (b) the capabilities we expect graduates to have. We then discuss the status and pace of institutional response to the development of the computing disciplines. After summarizing the relationship between degree programs and professional career opportunities, we summarize the elements that are common to all computing degree programs.

3.1. Curriculum summaries: A tabular view of computing degree programs

Snapshots are good for conveying information at a glance but, by their nature, they are incomplete in detail and can give incorrect impressions. In this section, we provide an expanded comparison for those who want to see more detail about the computing disciplines.

Table 3.1 provides a comparative view of the emphasis of study among the five kinds of computing degree programs we already discussed. The left column contains a list of some fifty topics that represent areas of knowledge and skill that students study in computing degree programs. This list approximates a union of the topics specified in the five major computing curriculum reports, providing a summary of the topics studied at the undergraduate level in one of more of the computing disciplines. If you are unfamiliar with them, you may consult the glossary of topics provided in Appendix A. The various curriculum reports sometimes use slightly different language for a given topic. They also differ in the degree to which they break down a topic into subtopics. As a result, the list of topics provided in Table 3.1 is not an exact match with the topic list of any of the curriculum reports. Rather, it is a summary of topics specified in the undergraduate computing curriculum reports.

For each item in this list of fifty topics, the other columns show numerical values for each of the five kinds of computing degree programs. These values range between 0 (lowest) and 5 (highest), and represent the relative emphasis each kind of computing degree program places on each given topic. For each of the five kinds of degree programs, each topic contains two values: one in the "min" column and one in the "max" column.

- The "min" value represents the minimum emphasis placed on that topic as specified in the curriculum report for that computing discipline. The "min" value indicates a discipline's minimum requirements relative to the minimum requirements of the other disciplines.
- The "max" value represents the greatest emphasis that one can reasonably place on the topic within the latitude provided by the curriculum report for that degree. Each discipline permits students some latitude in choosing an area of specialization and requires that a student's program of study go beyond the minimums defined in the curriculum report. It also permits each institution to establish requirements greater than those defined in the five curriculum reports. The "max" value indicates what one might reasonably expect of those who concentrate on the topic within the limits implied by other degree requirements.

Table 3.1: Comparative weight of topics across the five kinds of computing degree programs

Knowledge Area	***CE***		***CS***		***IS***		***IT***		***SE***	
	min	***max***	***min***	***max***	***min***	***max***	***min***	***max***	***min***	***max***
Programming Fundamentals	4	4	5	5	2	4	1	3	5	5
Algorithms and Complexity	2	4	5	5	1	2	0	1	4	4
Computer Architecture and Organization	5	5	2	4	1	2	1	2	2	4
Operating Systems Principles & Design	2	4	3	5	1	1	1	1	3	4
Operating Systems Configuration & Use	2	3	2	4	2	3	5	5	2	4
Net Centric Principles and Design	1	3	2	4	1	3	3	4	2	4
Net Centric Use and configuration	1	2	2	3	2	4	5	5	2	3
Theory of Programming Languages	1	2	3	5	0	1	0	0	2	4
Human-Computer Interaction	2	5	2	4	2	5	4	5	3	5
Graphics and Visualization	1	3	1	5	1	1	0	0	1	3
Intelligent Systems (AI)	1	3	2	5	1	1	0	0	0	0
Information Management (DB) Theory	1	3	2	5	1	3	1	1	2	5
Information Management (DB) Practice	1	2	1	4	4	5	2	4	1	4
Scientific computing (Numerical mthds)	0	2	0	5	0	0	0	0	0	0
Organizational Theory	0	0	0	0	1	4	1	2	0	0
Management of Info Systems Organ'tion	0	0	0	0	3	5	0	0	0	0
Decision Theory	0	0	0	0	3	3	0	0	0	0
Organizational Behavior	0	0	0	0	3	5	1	2	0	0
Organizational Change Management	0	0	0	0	2	2	0	1	0	0
Legal / Professional / Ethics / Society	2	5	2	4	2	5	2	4	2	5
General Systems Theory	0	0	0	0	2	2	1	3	0	0
Information Systems Development	0	2	0	2	5	5	1	3	2	4
Risk Management (Project, safety risk)	2	4	1	1	2	3	1	4	2	4
Project Management	2	4	1	2	3	5	1	3	4	5
Analysis of Business Requirements	0	1	0	1	5	5	1	2	0	2
Engineering Foundations for SW	1	2	1	2	1	1	0	0	2	4
Engineering Economics for SW	1	1	1	1	2	2	1	1	2	2
Software Modeling and Analysis	1	3	2	3	3	3	0	0	4	5
Software Design	2	4	3	5	1	3	1	1	5	5
Software Verification and Validation	1	3	1	2	1	2	0	0	4	5
Software Evolution (maintenance)	1	3	1	1	1	2	1	1	2	4
Software Process	1	1	1	1	1	1	1	1	2	4
Software Quality	2	3	2	3	1	2	1	1	3	4
Comp Systems Engineering	5	5	1	2	0	0	0	0	2	3
Embedded Systems	2	5	1	3	0	0	0	0	2	4
Circuits and Systems	5	5	0	2	0	0	0	0	0	0
Electronics	5	5	0	0	0	0	0	0	0	0
Digital logic	5	5	2	3	1	1	1	1	1	3
E-business	0	0	0	0	4	4	1	1	0	3
Distributed Systems	3	5	1	3	2	4	1	3	2	4
Digital Signal Processing	3	5	0	2	0	0	0	0	0	2
VLSI design	2	5	0	1	0	0	0	0	0	1
HW testing and fault tolerance	3	5	0	0	0	0	0	2	0	0
Security: issues and principles	2	3	1	4	2	3	1	3	1	3
Security: implementation and mgt	1	2	1	3	1	3	3	5	1	3
Systems administration	1	2	1	1	1	3	3	5	1	2
Systems integration	1	4	1	2	1	4	4	5	1	3
Digital media development	0	2	0	1	1	2	3	5	0	1
Technical support	0	1	0	1	1	3	5	5	0	1
Mathematical foundations	5	5	4	5	2	4	2	4	4	5
Interpersonal communication	3	4	1	3	3	5	2	4	3	4

Again, "min" represents the minimum called for by the curriculum guidelines, and "max" represents the greatest emphasis one might expect in the typical case of a student who chooses to undertake optional work in that area or who graduates from a school that chooses to require its students to achieve mastery beyond that required by the curriculum reports.

Both "min" and "max" values refer to reasonable expectations for the general case. Obviously, for any individual student or degree program, the "min" value might be as low as 0 and the "max" value might be as high as 5, irrespective of the prevailing curricular requirements for each discipline.

3.1.1. What the tabular comparison represents

Table 3.1 represents the consensus of judgment reached by the CC2004 Joint Task Force. It formulated that consensus from an examination of the discipline-specific body of knowledge found in the most recent curriculum report for each of the computing disciplines: *computer engineering, computer science, information systems, information technology,* and *software engineering*.

It used the results of that examination to define the topical elements of the table. It also heavily influenced the numerical values assigned to each topic for each discipline. The discipline-specific bodies of knowledge provide some quantifiable data concerning the minimum coverage called for by each discipline for each topic. However, they do not provide information that is sufficient to permit any useful formal calculation of the relative emphasis each discipline places on a given topic.

To obtain useful indications of the relative emphasis each discipline places on a given topic, it is necessary that we apply our best judgment as to how to integrate various hard and soft factors into a meaningful metric. Hard factors are the numerical specifications found in the discipline-specific bodies of knowledge. Soft factors include:

- Differences in the perspective endemic to the various computing disciplines. Each of the five computing disciplines has its own unique perspective and agenda, as characterized in Sections 2.2 and 2.3, above.
- Differences in the meanings attached to seemingly identical terms. While each computing discipline includes coverage of various computing topics, the existence of unique discipline-specific perspectives has the effect of attaching different meanings to the same term. For example, while each discipline requires study of programming fundamentals, networking, operating systems, etc., the precise set of knowledge and skills associated with these topics varies by discipline. This issue is addressed more fully Sections 3.2 and 3.3, below.
- Differences in the latitude available to study optional topics. Despite efforts to make each of the five discipline-specific curriculum reports international in scope, many are implicitly oriented to the American system of defining undergraduate degree programs. In the American system, the amount of technical study that can be required for a degree program in most subjects is significantly limited. Such limitations affect all computing degree programs equally. In addition to this generic constraint, some computing degree programs must devote a greater portion of their limited resources to a fixed agenda, which in turn limits how much freedom is available to study computing topics that are not required for the degree. For example, degree programs in *computer engineering* must devote significant study to topics required by (a) the engineering profession, and (b) the CE emphasis on hardware-related topics. This curtails how much study CE students may devote to optional computing topics. Degree programs in *information systems* face similar limitations due the requirement that IS students study business and organizational topics. The fact that not all computing disciplines have equal flexibility places constraints on degree programs that are difficult to translate in any precise or formal way into our schema of "min" and "max" topical coverage.

Due to the range and nature of these factors, the numerical values assigned in Table 3.1 reflect a synthesis of hard and soft factors constructed by the CC2004 Task Force.

3.1.2. Using the table: Two related examples

To see how to use the table provided in Table 3.1, we will consider the fourth and fifth topics in the table: "Operating Systems - Principles and Design" and "Operating Systems - Configuration and Use". Both topics concern operating systems. A reader who is not familiar with the terminology can consult the glossary in Appendix A:

- Readers who are unsure what "operating systems" are can learn from the glossary. The term "operating system" refers to a specific kind of software (such as Windows, Linux, UNIX, Mac OS, etc.) which permits the human user to interact with a computer. It also enables a computer to manage its resources (memory, disk drives, monitor screen, network interface, etc.) so that it can run whatever application programs (word processor, spreadsheet, web browser, etc.) the user asks the computer to run. In short, an operating system is software that runs "in the background", permitting the computer to operate in useful ways.
- A reader can also learn from the glossary that the former topic, "Operating Systems - Principles and Design", refers to foundational knowledge that enables a student to understand the tasks an operating system must perform. It involves the various strategies and tactics that an operating system might deploy do those things, the kinds of mechanisms that the operating systems designer can use to implement those strategies and tactics, the strengths and weaknesses of various popular approaches, and so on. In addition, we have the expectation that a student might complete a major programming project, either creating an operating system "from scratch" or creating a significant enhancement to an existing operating system.
- Similarly, the reader can learn from the glossary that the latter topic, "Operating Systems - Configuration and Use", is concerned with the practical mastery of the capabilities of mainstream operating system products. Rather than focus on the underlying concepts and principles for designing and implementing operating systems, this topic focuses on developing the student's ability to make full use of the various capabilities provided by existing operating systems. The goal is produce students who know the strengths and limitations of two or more mainstream operating systems, know how those attributes relate to both organizational policy and individual user needs, and know how to use operating systems capabilities to satisfy user needs and implement organizational policy.

The reader can consult Table 3.1 to see how the various disciplines compare in the emphasis they place on each of these two topics:

- For the former topic, "Operating Systems – Principles and Design", we see that both IS and IT programs give it less emphasis than do CE, CS, and SE programs. For typical degree programs in IS and IT, both the "min" and "max" values are "1", which indicates that students in those programs are exposed to some basic concepts and terminology but typically do not study OS principles and design in any substantive way. In contrast, CE, CS, and SE programs have higher "min" and "max" values, indicating that they have higher minimum standards and a higher ceiling for student mastery. The fact that a higher "min" value is shown for CS and SE (3 each) programs than is shown for CE (2) indicates that the CS and SE curricula typically feature more coverage of OS principles and design than do CE programs. The fact that a higher "max" value is shown for CS (5) than for CE and SE (4 each) indicates that CS curricula typically have more room in their program of study to permit in-depth coverage of OS principles and design for those who desire it.
- For the latter topic, "Operating Systems – Configuration and Use", we see a different pattern of relative emphasis among the degree programs. While all degree programs provide substantive experience in using and configuring operating systems, IT programs have the highest "min" and

"max" values (5 each). This indicates that IT programs focus their coverage of OS explicitly on configuration and use. It also indicates IT programs expect all their students to obtain significant capability in this area. The other degree programs have weaker minimum expectations that are comparable to each other. There is slightly more room in CS and SE programs for optional mastery of this topic than in CE and IS programs, but all place less emphasis on OS configuration and use issues than do IT programs.

The comparison provided in Table 3.1 allows us to conclude that a student who wants to understand the principles and design of operating systems will not be well served by IS and IT programs, will be better satisfied by CE, CS, or SE programs, and will have the greatest opportunity for in-depth study in CS programs. In contrast, a student who is interested primarily in the practical configuration and use of operating systems will be best served by an IT program, as each of the other degree programs provide a substantive foundation in practical OS use but less opportunities for in-depth mastery in this area. A student wishing to pursue both OS topics would likely gravitate toward a CS or SE degree program, where he or she will sacrifice depth with respect to practical application to obtain a better balance of principles and application.

3.2. Degree outcomes: Comparing expectations of program graduates

The previous section provides a comparative view of the emphasis of study in the major kinds of computing degree programs. This section provides a comparative view of the performance capabilities expected of the graduates of each kind of degree program. While the previous section summarizes what a student will study, this section summarizes the expectation of the student after he or she graduates and begins a career.

Table 3.2 lists 70 performance capabilities across some 11 categories. For each capability, each discipline is assigned a value from 0 to 5. The value 0 represents no expectation whatsoever and the value 5 represents the highest relative expectation. It shows the particular ways in which, in general terms:

- *Computer engineers* should be able to design and implement systems that involve the integration of software and hardware devices.
- *Computer scientists* face expectations that range from theoretical work to software development.
- *Information systems* specialists should be able to analyze information requirements and business processes and they should be able specify and design systems that are aligned with organizational goals.
- *Information technology* specialists face high expectations with respect to using technology and to the planning, implementation and configuration of computing infrastructure.
- *Software engineers* should be able to design and implement large-scale software systems.

In contrast to Table 3.1, which summarizes the *inputs* provided to students by degree programs, Table 3.2, focuses on *outputs*, summarizing the relative capability expectations of graduates.

Table 3.2. Relative Performance Capabilities of Computing Graduates by Discipline

Area	*Performance Capability*	*CE*	*CS*	*IS*	*IT*	*SE*
Algorithms	Prove theoretical results	3	5	1	0	3
	Dev. solutions to prog'ing problems	3	5	1	1	3
	Develop proof-of-concept programs	3	5	3	1	3
	Determine if faster solutions possible	3	5	1	1	3
Application infrastructure	Manage web sites	2	2	4	5	2
	Create e-commerce software	2	3	4	5	4
	Create multimedia systems	2	3	4	5	3
	Develop health related applications	3	3	3	2	5
	Create e-learning systems	1	2	5	5	3
	Develop business applications	1	3	5	4	3
	Evaluate new forms of search engine	2	4	4	4	4
Application programs	Design a word processor	3	4	1	0	4
	Use word processor features well	3	3	5	5	3
	Train and support word processor	2	2	4	5	2
	Design a spreadsheet	3	4	1	0	4
	Use spreadsheet features well	2	2	5	5	3
	Train and support spreadsheet users	2	2	4	5	2
Computer programming	Do small-scale programming	5	5	3	3	5
	Do large-scale programming	3	4	2	2	5
	Do systems programming	3	4	1	2	4
	Develop new software systems	3	4	3	1	5
	Create safety-critical systems	4	3	0	0	5
	Manage safety-critical projects	3	2	0	0	5
Hardware and devices	Design embedded systems	5	1	0	0	1
	Implement embedded systems	5	2	1	1	3
	Design computer peripherals	5	1	0	0	1
	Design complex sensor systems	5	1	0	0	1
	Design a chip	5	1	0	0	1
	Program a chip	5	1	0	0	1
	Design a computer	5	1	0	0	1
Human-computer interface	Create a software user interface	3	4	5	4	4
	Produce graphics or game software	2	5	0	0	5
	Design a human-friendly device	4	2	0	1	3
Information systems	Define information system	2	2	5	4	2
	Design information systems	2	3	5	4	3
	Implement information systems	3	3	4	3	5
	Train users to use information systems	1	1	4	5	1
	Maintain and modify information	3	3	5	5	3
Information management (Database)	Design a database system	2	5	1	0	4
	Use a database system	2	2	5	5	2
	Implement information retrieval	1	5	3	3	4
	Select database products	1	3	5	5	3
	Configure database products	1	2	5	5	2
	Manage databases	1	2	5	5	2
	Train and support database users	2	3	5	5	3
IT resource planning	Develop corporate information plan	1	1	5	3	1
	Develop computer resource plan	2	2	5	5	2
	Schedule/budget resource upgrades	2	2	5	5	2
	Install/upgrade computers	4	3	4	5	3
	Install/upgrade computer software	3	3	4	5	3
Intelligent systems	Design auto-reasoning systems	2	4	0	0	2
	Implement auto-reasoning systems	2	4	0	0	5
	Implement intelligent systems	4	4	1	0	4
	Implement information retrieval	2	3	5	3	4
Networking and communications	Design network configuration	4	4	3	2	3
	Select network components	2	2	4	5	2
	Install computer network	2	1	3	5	2
	Manage computer networks	3	3	3	5	3
	Implement communication software	5	4	1	0	4
	Manage communication resources	1	0	3	5	0
	Implement mobile computing system	5	3	0	0	3
	Manage mobile computing resources	3	2	2	4	2

3.3. International considerations

The CC 2001 activity has benefited from international input. This provides assurances that the best advice was being used t guide developments and this in turn gives confidence about the international competitiveness of graduates both in terms of their skills and of their outlook. Of course, in turn the *Computing Curricula* reports serve as important points of reference for degree programs in other countries.

In seeking to provide advice and illustrations that would have international appeal we can claim some success and some interesting points of comparison arose. Very naturally there are differences in the structure of the academic year, in the emphasis given to the study of computing within a degree program (in the UK, for example, almost all classes will be oriented toward computing), in the quality control mechanisms, and so on. In addition to these it was noted that:

- In the US, there is a very strong sense of a core for each discipline. The intent of this is to capture those elements of a discipline considered fundamental and which all students of that discipline should fully understand. This has benefits in terms of understanding the abilities of graduates and it facilitates transfer between institutions. The argument and agreement over core tends to result in the emergence of strong discipline definitions and clear understanding of the degree titles
- The core idea is far less prominent in countries such as the UK, where degree titles function as strong marketing opportunities. As a result, by US standards, a vast number of degrees exist with a rich variety of titles that reflect different emphases and different career opportunities. In this context, guidance and quality issues associated with degrees are taken care of by criteria such as those outlined in Section 3.6

3.4. The pace of change in academia: Disciplines and available degrees

As discussed previously, the landscape of the computing disciplines has undergone dramatic change because of the explosion of computing during the 1990s. The computing field has evolved to the point that we are seeing the emergence of degree programs focused on the challenges that now confront both the computing profession and our computing-dependent society as a whole.

At many colleges and universities, the evolution of computing discussed in Chapter 2 does not reflect the kinds of degree programs offered. This is because it is the nature of many institutions to be conservative, and because the complex nature of academic degree programs means that it is difficult to implement significant changes rapidly. Thus, at some colleges and universities the choices of computing degree programs still looks more like the “pre-1990s” view shown in the top portion of Figure 2.2 than the “post-1990s” view shown at the bottom of Figure 2.2. This discrepancy between the current landscape of computing disciplines and the offerings at a particular college or university is a reflection of the fact that the pace of change in computing is quite rapid while the pace of change in academic institutions generally is quite slow.

This natural institutional lag can create problems for students who are trying to choose a computing-related degree program that fits their interests and goals. It can also create problems for educators who are trying to ensure that their educational programs are providing up-to-date programs that serve their constituents.

We can make an important observation. Looking at the development of computing over the last twenty or so years, one can observe that there has been a dramatic shift in emphasis towards interaction (and perhaps away from the study of the algorithm). We can view the move towards interaction as an important mark of success; it highlights the fact that people are involved in using and interacting with them far more than in the early days. It is very natural therefore that new programs of study should emerge to reflect that. The emergence of IT ands IS programs is part of that. Indeed, there is scope for an even richer variety of possibilities.

The Joint Task Force of this report estimates that as of May of 2004, the approximate number of programs in existence for each discipline worldwide is approximately as follows: 800 for computer engineering programs, 4000 for computer science programs, 200 information technology programs, and 200 software engineering programs. For information systems, the question is more complicated because degree programs associated with the information systems agenda have many different names; our best estimate is that approximately 2000 information systems programs exist. The numbers above are subject to verification. However, they do provide a measure of difference in the size and scope of each discipline. In addition, many computing-related degree programs use other names for their degrees. A few of these programs focus on special niches in computing; others are hybrids that combine features of the major kinds of degree programs.

A key issue in terms of institutional lag may lie in the ability of readers of this report to recognize it. In very general terms, there are usually metrics or marks of quality that we tend to look for. For instance

- Involvement of industrialists in advisory boards and importantly students in committees that vet programs
- Quality control mechanisms operated by institutions and these really ought to involve external experts
- Employment statistics of graduates which will be related to the reputation of graduates and of the institution
- Accreditation of degrees provides a mark of quality and will address issues such as the currency of the program of study

Beyond this, there are additional comments we can make, which vary across the five the major computing disciplines. We summarize these below.

3.4.1. Computer engineering

The change from the "pre-1990s view" of computing to the "post-1990s view" involves the emergence of *computer engineering* from within *electrical engineering*. Typically, both kinds of programs exist in the same academic department. Thus, the change we described earlier is an evolutionary change in program stature and emphasis, not the creation of a new computing discipline. Although not true on the international stage, in the US there has always been a single home for those who study computer hardware. It used to be electrical engineering. It has become computer engineering. At many universities, the change from the "pre-1990s" to "post-1990s" view has already taken place. However, even some of the best schools do not offer a separate computer engineering program. At some schools, electrical engineering and computer science reside in the same school, and computer engineering becomes a natural merging of interests among faculty in those two disciplines.

There are notably fewer programs in *computer engineering* than in *computer science* or *information systems*. In the US, this is because most American colleges do not offer engineering programs of any kind. American engineering programs reside only in relatively large universities that can meet the special requirements of the professional engineering community. We do not expect this to change significantly in the future.

3.4.2. Computer science

There are far more degree programs in computer science than in any other computing discipline. Almost all colleges and universities offer a CS degree. To some extent, this is an historical artifact: *computer* *science* was the only substantive computing discipline that focused explicitly on software development when academic computing degree programs emerged in the 1970s. When most colleges created their computing degree programs, computer science was the only choice that had strong ties to math, science, and/or engineering. (IS programs developed around the same time, but their primary ties were to business schools.)

The increased diversity we see in the post-1990s computing disciplines is fairly localized in areas that affect computer science. The new computing disciplines target career areas that graduates of CS programs traditionally filled. Because there are more programs in computer science than in any other computing discipline, and because the newer computing disciplines target students who would otherwise likely be computer science majors, computer science, perhaps, has the most to lose as new computing disciplines emerge.

Currently, an ongoing discussion exists regarding the relationship between what *computer science programs teach* and what most *graduates of computer science actually do* in their careers. To understand this discussion, it is necessary to review the characterization of computer science provided in Section 2.3.2. The work of computer scientists falls into three categories: developing effective ways to solve computing problems; devising new ways to use computers; and designing and implementing software. Let us consider what is involved in a career path in each area:

- *Career Path 1: Developing effective ways to solve computing problems*. This refers to the application of computer science theory and knowledge of algorithms to ensure the best possible solution of computationally intensive problems. As a practical matter, a career path in this area typically requires graduate work to the Ph.D. level, followed by a position in a research university or industrial R&D laboratory.
- *Career Path 2: Devising new ways to use computers*. This refers to innovation in the application of computer technology. A career path in this area can involve advanced graduate work, typically to the Ph.D. level, followed by a position in a research university or industrial R&D lab, or it can involve entrepreneurial activity such as was evident during the "dot com" boom of the 1990s, or it can involve a combination of the two.
- *Career Path 3: Designing and implementing software*. This refers to the work of software development. Nowadays we might interpret this to include aspects such as web development, middleware, security issues, and mobile computing. The majority of computer science graduates choose this career path. While a bachelor's degree is generally sufficient for entry into this kind of career, many software professionals return to school to obtain a terminal master's degree. (Rarely is a doctorate involved.) Career positions occur in a wide variety of settings, including large or small software companies, large or small computer services companies, and large organizations of all kinds (industry, government, banking, healthcare, etc.). The discipline of software engineering also educates its students for this career path.

Computer science programs intend to prepare students for these three career paths. In addition, the following is a fourth career path CS programs do not target but nonetheless draws many computer science graduates.

- *Career Path 4: Planning and managing organizational technology infrastructure.* This refers to the kind of work for which IT might also educate students, with an emphasis on business.

Of these four career paths, the first two are important elements of the identity of *computer science* and are the kind of career paths that many computer science faculty wish to see their students choose. As a practical observation, only a small minority of the students who earn computer science degrees chooses them. For these first two career paths, a strong program in computer science is clearly the preferred choice.

The latter two career paths are the focus of debate. These careers draw the overwhelming majority of computer science graduates. They are also the focus of new computing disciplines (software engineering and information technology, respectively), which have come into being as an additional and more focused alternative to computer science programs at preparing students for these career paths. In addition, a significant number of graduates in information systems has over the years selected organizational roles that are very similar to these career paths. As yet, no resolution has occurred to the debate about the relative value of computer science programs vs. software engineering and information technology programs. However, the issues that arise in the debate are well defined. Because software engineering and information technology programs have different goals, the issues discussed are somewhat different. We discuss the relevant issues viz. computer science in the subsequent subsections devoted to software engineering and information technology*,* respectively

As computer science programs face growing competition from degree programs in software engineering and information technology*,* students interested in the latter two career paths will have more choices than ever before. For the latter two careers paths, we should evaluate the relative value of computer science programs in the light of emergence of degree programs in the two new computing disciplines.

3.4.3. Information systems

The difference between the "pre-1990s" and "post-1990s" role of information systems concerns the expanded role of information technology in organizations of all kinds. Historically, information systems programs prepared students to work in the business world. Their preparation included a foundation in accounting, finance, and other core business areas. On the technology side, IS students could expect to become familiar with computer applications related to these traditional business areas, especially database management systems, and with spreadsheets and other off-the-shelf software products that have broad utility to business people. In addition, they learned how to develop functionally oriented applications.

In the "post-1990s" world, an up-to-date IS program is focusing on the broader role of IT-enabled information utilization and business processes for a wide range of organizations. What information does the enterprise need? How do we generate that information? Does the information go to the people who need it? Do we present the information to them in ways that allow them to use it? Have they structured the organization correctly to use technology in the efficient ways? Have they properly designed the business processes of the organization? Does the organization fully utilize the communication and collaboration capabilities of information technologies? Is the organization capable of adapting quickly enough to changing external circumstances? These are the important issues that organizations increasingly rely on IS people to address.

Thus, institutional lag is relevant to an IS program insofar as we consider whether it has expanded its focus beyond just the "pre-1990s" focus on narrowly defined functional business processes. That traditional role is still there, but it is no longer sufficient. The meaningful question may be, "Has an IS program broadened its scope to include an integrated view of the organization with complex information needs and high-level dependency on IT-enabled business processes?" Web-based distributed technologies provide infrastructure for globally connected and ubiquitously accessible organizations, and modern IS programs have to address the needs of such organizations. IS students must learn how to assess and evaluate organizational information needs, specify information requirements, and design practical systems to satisfy those requirements. If a program instead focuses only on the design and development of narrow functional applications and the use of personal productivity tools, it is seriously behind mainstream IS programs.

In addition to these concerns, IS faces challenges from the emergence of IT programs, much as CS programs do. Traditionally, many graduates of IS programs have functioned in roles that are similar to the roles for which IT programs explicitly prepare their students. As the number of IT programs grow, many IS departments will have to reevaluate how to serve and define their core audience.

3.4.4. Information technology

In the last few years, degree programs in information technology have emerged and developed to such an extent that they now should be an important part of any discussion of computing degree programs. As summarized in Section 2.3.4, IT programs exist to produce graduates who know how to make information technology work in a wide range of settings. Organizations of all kinds have become dependent on networked computing infrastructure to such an extent that they cannot function without that infrastructure. IT people are prepared to select, manage, and maintain that infrastructure, ensuring that it meets organizational needs. They also create digital content for that infrastructure and take care of IT-support needs of individuals who use it.

The emergence of IT programs represents a grass-roots movement by computing educators to respond to the very real needs of both their local communities and their students. IT programs exist, not because computer science or information systems programs failed to "do their job", but because those disciplines each define themselves as having a different job. IT programs exist because nobody was seriously trying to do the job that IT programs are doing.

Only a few years ago, computing educators in the US were not familiar with IT degree programs, although such programs have existed for many years elsewhere. Today, many such programs are emerging and we expect to see this number increase further in the years to come. It was not until 2001 that college-level IT educators began to organize, and in the short period since, they have formed a professional organization, held several conferences, and have made substantive progress in developing curriculum and accreditation guidelines for IT degree programs. It is no exaggeration to say that IT programs have exploded onto the scene in the US.

Some people question whether IT programs are a passing fad. Others ask if IT programs are too technical in nature to deserve the status of an academic discipline. People asked similar questions about computer science more than thirty years ago. Yet within a few years, virtually all colleges began to offer CS degrees. We expect to see similar results with respect to IT. They address an important need that is a widespread throughout society. To the extent that organizations rely on computer technology, the IT discipline has a key role to play.

There are two important issues here:

- *Rigor*: Planning and managing an organization's IT infrastructure is a difficult and complex job that requires a solid foundation in applied computing, as well as management and people skills. Those in the discipline require special skills – in understanding, for example, how to compose and structure software systems and to contrast their strengths and weaknesses. There are important software systems concerns such as reliability, security, effectiveness, and efficiency for their intended purpose, and importantly usability that are all vital. These topics are difficult and intellectually demanding.
- *Acceptance*: The IT discipline is "the new kid on the block" and, as a result, faces problems of acceptance among the more established disciplines. This is a natural phenomenon, and it will take time and experience for those in the more established computing disciplines to recognize the value that the IT discipline provides. As IT establishes itself as a discipline with its own intellectual core, a rigorous curriculum and accreditation guidelines, acceptance and respect will come.

In the short term, these two issues combine to present a potential danger. At many institutions, administration is motivated to see an IT program created to respond to community needs and to provide more choices for prospective students. Whenever an institution creates an IT program, it must take special care to ensure that it implements the program properly. One must ensure that the people who are responsible for an IT degree program recognize IT's importance and are excited about creating high-quality educational experiences for IT students.

3.4.5. Software engineering

The development of SE is a response from within the CS community to a very real problem. While CS programs have shown that they can produce students who have sound skills in programming fundamentals, many believe that they have not been successful at reliably producing graduates who can properly understand and develop software systems; in the extreme they have to engineer rigorously large software programs. In the post-1990s world, many software projects are large and complex, and there is a pressing need for software engineers who can apply professional practices to ensure that the software is reliable and that they produce it on schedule and within budget. We know that major software companies have found it necessary to develop their own in-house educational programs to compensate for the capability shortcomings of CS graduates. SE programs represent an effort from within CS to make the undergraduate experience more successful at providing students with an adequate set of knowledge and skills for careers as software professionals.

As a practical matter, CS and SE degree programs have much in common. Both disciplines recognize that the subject matter of computing has grown to the point where no single degree program can expect its students to grasp the entire field. Thirty years ago, it was reasonable to expect CS undergraduates to "study everything"; now, there is too much of it to fit in a four year (or even five year) program of study.

Both CS and SE curricula begin by requiring a foundation in programming fundamentals and basic CS theory. They diverge in what they focus on beyond those core elements. CS programs tend to keep the core small and then expect students to specialize selectively in one or two of areas of CS concentration (such as systems, networking, database, artificial intelligence, theory, etc.). In contrast, SE programs generally provide less freedom of choice about advanced computer science topics, and instead expect students to focus on a range of topics that are essential to the SE agenda (problem modeling and analysis, software design, software verification and validation, software quality, software process, software management, and so on). In addition, while both CS and SE programs typically require students to experience team project activity, SE programs tend to involve the students in significantly more of it, as effective team processes are essential to effective SE practices. A key requirement from the SE report is that SE students should learn how to build software that is *genuinely* useful and usable by those from another discipline.

Two questions about how SE programs will develop remain unanswered:

- *To what extent will degree programs in SE emerge from within CS departments as a side-by-side alternative to the traditional CS degree?* Some disagreement exists in this respect surrounding the nature and amount of rigorous SE experience that a robust undergraduate education should have. We can view the development of SE programs as an effort to broaden the appeal of computing degrees within an institution.
- *To what extent will the legal implications of using the word "engineering" force American colleges and universities to choose a different name for their SE degree programs?* The professional engineering community is protective of its identity and the word "engineering" looms large in that identity. Software engineering is fundamentally different from the other engineering disciplines (due to the intangible nature of software, and to SE's focus on human processes rather than the laws of physics, for example). Yet the traditional strength of engineering (robust methods for creating reliable artifacts) is at the core of the SE agenda. It will take some time for the American engineering community itself to recognize how SE is fundamentally different from other engineering disciplines, and perhaps even longer for reason to prevail about the use of the word "engineering". In the U.K., the computing and engineering communities came to a sensible understanding about this issue

decades ago. America lags behind in this case, as it has yet to reach a reasonable accommodation to the situation.

It appears that most CS students anticipate that their professional careers will involve doing software development work, while relatively few see their future on the research and innovation side of CS. We expect that, if substantive SE degrees were available as an option for CS students, many (perhaps most) CS students would select it. At present, that choice is not widely available. There are now about 20 SE degree programs in America. We expect this number to increase steadily, although not at the same explosive rate that we have seen from IT programs.

Within CS degree programs, the robustness of SE education varies widely. Most CS degree programs provide one or more SE courses. In some CS programs, SE courses are optional; in many others, they are mandatory. In addition, some CS programs offer SE as one of several areas of CS concentration. When evaluating CS programs that provide such an option, it is important to look closely at how rigorous that option is. Some faculty feel that taking one or two SE is sufficient; others believe that one or two courses cannot offer the kind of professional preparation that a dedicated SE program can.

3.5. The pace of change in the workplace: Degrees and career opportunities

Just as it takes time for academic institutions to adapt, it also takes time for industry and other players in the computing workplace to do the same. As a result, most job opportunities will reflect the "pre-1990s" model shown at the top of Figure 2.3. In other words, because new computing disciplines are new, no one yet expects people to have those qualifications.

With respect to CE, the new disciplines are not much of an issue. In both the "pre-1990s" and "post-1990s" landscape, it is clear where people go for expertise in hardware and related software. During the 1990s, CE adopted this role from EE; the change is largely complete. CE, like EE, is part of the professional engineering community, and this has implications for where one can earn a CE degree. Not every university has an engineering program. Employers looking for people with a CE background want graduates of an accredited CE program.

It is also clear where people look for expertise at the interface between the information needs of business and computing. IS occupies that role in both the "pre-1990s" and "post-1990s" landscape. It is essential that those who pursue this path make sure that they find an IS program that continues to update its vision and has an emphasis that fits the interests of the prospective student. For IS, the situation is complicated by the range of names such programs go by.

The new computing disciplines (SE and IT) raise questions primarily for people who would otherwise major in CS (viz. both SE and IT) or IS (viz. IT). At present, most people who function as serious software engineers have degrees in CS, not in SE. In large part, this is because CS degrees have been available for 30 years and SE degrees have not. The story is similar for people working in the IT profession: most come from CS or IS programs. It is far too soon for someone who wants to work as a software engineer or as an information technology practitioner to be afraid that they won't get the chance to do so if they don't graduate from a degree program in one of those new disciplines. In general, a CS degree from a respected program is still the most flexible of degrees, as it can open doors into the professional worlds of CS, SE, IT, and (in some cases) CE. Similarly, a degree from a respected IS program permits entry into the worlds of IS and IT.

This situation presents ambiguity to students who are trying to determine what discipline to study and to educators who are trying to decide how to serve their constituents. We identify some factors that educators should consider in Chapter 4. We focus on the factors students should consider in the *Guide*, which is the second part of this report and is also available as a standalone document.

3.6. Shared identity: Common requirements of computing degrees

As we have seen, each of the major computing disciplines has its own character. Each one is somewhat different from its siblings in emphasis, goals, and capabilities of its graduates. At the same time, they have much in common. Therefore, any reputable degree program in computing should include each of the following elements.

1) Essential and foundational underpinnings of the discipline. These may be abstract, e.g., formal theory rooted in mathematics, or they may be applied, e.g., professional values and principles; often, they will include both abstract and applied elements. Regardless of their form or focus, the underpinnings must highlight those essential aspects of the discipline that remain unaltered in the face of technological change. The discipline's foundation provides a touchstone that transcends time and circumstance, giving a sense of permanence and stability to its educational mission. Students must have a thorough grounding in that foundation.

2) Foundation in the concepts and skills related to computer programming. There are three layers to this foundation:

   a) An intellectual understanding of, and an appreciation for, the central role of algorithms;

   b) Fundamental programming skills to permit implementation of algorithms in software;

   c) Software engineering principles and technologies to ensure that software implementations are robust, reliable, and appropriate for their intended audience.

3) Understanding of the possibilities and limitations of what computer technology (software, hardware, and networking) can and cannot do. This foundation has three levels:

   a) An understanding of what current technologies can and cannot accomplish;

   b) An understanding of the limitations of computing, including the difference between what computing is inherently incapable of doing vs. what may be accomplished via improved computer technology in the future;

   c) The impact on individuals, organizations, and society of deploying technological solutions and interventions.

4) Understanding of the concept of the lifecycle, including the significance of its phases (planning, development, deployment, and evolution), the implications for the development of all aspects of computer-related systems (including software, hardware, and human computer interface), and the relationship between quality and lifecycle management.

5) Understanding of the essential concept of process, in at least two meanings of the term:

   a) Process as it relates to computing, especially program execution and system operation;

   b) Process as it relates to professional activity, especially the relationship between product quality and the deployment of appropriate human processes during product development.

6) Study of advanced computing topics that permits students to visit and understand the frontiers of the discipline. This is typically be accomplished via explicit inclusion of learning experiences that lead students from elementary topics to advanced topics or themes that pervade cutting-edge developments.

7) The identification of and acquisition of skill sets that go beyond technical skills. Such skill sets include interpersonal communication skills, team skills, and management skills as appropriate to the discipline. To have value, learning experiences must build such skills (not just convey that they are important) and teach skills that are transferable to new situations.

8) Exposure to an appropriate range of applications and case studies that connect theory and skills learned in academia to real-world occurrences to explicate their relevance and utility.

9) Attention to professional legal and ethical issues such that students evidence attitudes and priorities that honor, protects, and enhances the ethical stature and standing of the profession.

10) Demonstration that each student has integrated the various elements of the undergraduate experience by undertaking, completing, and defending original work in a capstone experience.

Chapter 4: Conclusion

This chapter is under development.

Its content will alert the reader to issues and considerations as outlined below.

Its length should not exceed six pages.

4.1. Institutional considerations

4.2. Curricula and accreditation

4.3. Future considerations

Part Two

A Guide to Undergraduate Degree Programs in Computing

[*This 12-page Guide is under development.*]

3. Student Considerations

3.1. The Challenges and the Excitement of Computing

3.2. Career Opportunities

3.3. The impact of degree choice on your future

Appendix A

Glossary of Terms

[*Under development*]

References

[*Under development*]