Contemporary Capstone Computer Courses: Lessons From The Service Sciences

M. Keith Wright, PhD, CISA, PMP, University of Houston – Downtown, USA
Charles J. Capps III, DBA, SPHR, Sam Houston State University, USA

ABSTRACT

Enrollment in computer programming courses has plummeted in the past decade. Facing a similar situation in the 1960s, the mathematics community responded by inventing the "new math." Unfortunately the new math failed because it was too abstract for students to see connections with their lives, and because math teachers were not adequately prepared. Many of today's computing related degree programs are in danger of failing for similar reasons. This paper argues that, besides off-shoring; there may be other less obvious reasons for the drop in enrollment. These reasons include curriculums that overemphasize functional programming, and under-emphasize ethics and practical service internships. This paper further argues that modern curriculums for the fields of Computer Science, Information Systems, Information Technology, and Software Engineering could all be improved by viewing them as sub-specialties of the newly emerging discipline of Service Sciences. The paper concludes by sketching a basic curriculum for a hypothetical new program we call the School of Artificial Systems and Service Sciences. It is an extension of the philosophical approach used at Yale Medical School.

Keywords: programming curriculum, computer science, information systems, service sciences

INTRODUCTION

When Edsger Dijkstra wrote his seminal paper on programming, programmers were lost in millions of lines of spaghetti code [3]. Dijkstra showed the way out. Now programmers have lost their way again -- this time amidst thousands of unread resumes. As a result, in 1998 the percentage of all freshmen planning a major in computer science was only 1.4%. Between 2000 and 2004, the percentage of incoming computer science freshmen fell by 60%. Drop rates of 30-50% were common. Results are similar for the other computing related fields including Information Systems, Software Engineering, and Information Technology [6, 17]. In the preface of his book "Pre-calculus Mathematics in a Nutshell," Professor George F. Simmons wrote that the "New Math" of the 1960s failed because it produced students who had "heard of the commutative law, but did not know the multiplication table" [4]. A similar situation now exists for the computing related disciplines that Herbert A. Simon referred to collectively as the "Sciences of the Artificial" [16].

This loss of student interest in Simon's sciences of the artificial has been blamed on various factors including media portrayals of computing as relatively unglamorous, and an impression that computing requires extraordinary programming skills. Computing journals perpetuate these notions. For example, Martin [8] argues that in capstone computer science courses students working individually should design and code a 'real world' application. Martin cites his experience of writing code for his small business. In contrast, we believe strongly that to emphasize programming, real or imaginary, via a capstone course is, in this day and age, to waste the precious limited time professors have with their students. Accordingly this paper focuses on other ideas to improve curricula for the artificial service sciences.

This article is organized as follows. First, the paper presents personal experiences of one of the authors as a professional code writer for several Fortune 500 companies, including IBM, from 1975 to 2000. This experience suggests that programming skills have been but a small part of what students have needed historically. Next, evidence from current economic literature is cited reminding the reader that professional code writing is no longer

the high paying career it once was. Finally, borrowing from Herbert Simon, the Yale Medical School curriculum, and the Information Technology Infrastructure Library (ITIL) ideas are presented on how to redesign computing curriculums to better address modern economic and organizational needs.

WHAT IS "PROFESSIONAL" PROGRAMMING?

Before rejoining academia, the primary author, Dr. M. Keith Wright, spent over twenty years as a professional programmer and his experiences reflect many trends from the era. Originally drawn to the field by a college level love for programming, he began his first programming job in 1974. In those days, computers and compilers were unavailable to all except academics, business owners and their employees. The principal entry level requirement for professional programming was a technical college degree that included a couple of programming courses (Assembly, and either FORTRAN or COBOL). The essential qualities employers wanted were analytical ability, intellectual curiosity, and loyalty. Programming was a rare skill and programmers were in short supply. The primary author's first job was in Houston, Texas, at a startup company with a government contract to digitize highways and power lines, the beginnings of technology found today in Google Earth and Yahoo maps.

Computer centers in those days were populated with machines such as IBM 360s, Burroughs 3700s, and UNIVAC 1100s. Using these machines were some of the best recent college graduates, old time COBOL and FORTRAN programmers, and part time academics. Wright was part of a fifteen person programmer team assigned to write code for map plotter drivers from scratch. He was doing design and using analytical geometry techniques learned in college. "I was making good money. I was perceived as a first-rate programmer. I was in programmer heaven!"

Unfortunately, the heavenly period lasted only about a year. Then he learned his first and hardest professional programming lesson: professional code writing is primarily re-writing. Sadly, beginning in the second year, Wright began re-writing his code the first of what seemed an endless number of times -- each re-write conforming to a different deployment environment. He had to re-write the code of many of his former team mates who had moved on to higher paying jobs. Thus, for the first time in his career, Wright was faced with a huge dose of what Fred Brooks first described as accidental complexity [2].[1]

Because of the short supply of skilled professionals, programmers seldom stayed at a job more than a year or so. The lure of extra money was compelling. Wright soon changed jobs and learned another hard lesson: small programming teams are better than big teams. He sub-contracted with a Silicon Valley firm called Informatics and became part of a one hundred person team developing a shop-floor control system for the United Airlines maintenance operations center in San Bruno, California. The project was the beginning of current supply-chain technology. What Wright did not understand was that he made a mistake by asking to be a programmer on the team. It seemed programmers were restricted to duties such as implementing CICS (green) screens.[2] Alas, most of what he considered to be the interesting project tasks were, he was told, to be done by the systems analysts. These tasks included database design, the module decomposition, and coupling strategy.

Further exacerbating his boredom, Wright was told systems analysts had not yet completed their work. This meant there was nothing ready to program, and that nevertheless, he was to "look busy." It seemed that a mistake in the project's plan allowed programmers to be hired about three months before they were needed, resulting in thousands of suspect man-hours billed to United Air Lines.

After a few weeks Wright could not face another day of trying to "look busy" and began seeking another opportunity. He thought perhaps his next job should be as a systems analyst and went to work for a Virginia based company, Information Engineering Systems Corporation. There he was part of a team doing data modeling for a supply chain project for Meijer department stores data center in Grand Rapids, Michigan.[3] That job was fine, except Wright noticed the only consumers of the data models were the data modelers. The intended consumers, the

[1] A change in processor control, during expression evaluation.
[2] Customer Information Control System. IBM http://en.wikipedia.org/wiki/CICS
[3] http://www.meijer.com

programmers, were employed by a different sub-contractor, Keane. [4] It seemed a mistake in the service level agreement allowed the programmers to proceed before the data models could be developed. This bothered him and he moved on to his next job.

Wright became a programmer again, this time with Levi Strauss in San Francisco working with operations research specialists coding an interesting CAD-CAM optimization algorithm to build blue jeans with the least amount of fabric -- using an early just-in-time inventory system, doing design, and using clever mathematical tricks. Once again he was in heaven until the day (two years later) when Levi Strauss off-shored their entire supply chain service.

Wright's quest for interesting and honest computing work continued for the next decade, taking him back to academia, the Texas Workers Compensation Commission, National Semiconductor, Advanced Micro Devices, and finally IBM in 1998. By the time Dr. Wright joined IBM, he was equipped with a University of Texas PhD in Information Systems. He volunteered for a small Java programming team working on an interesting insurance fraud detection data mining application against the advice of almost all his new IBM colleagues who viewed programming as demeaning work.

Once again he was in programmer heaven. Wright was working with J2EE and learning web technologies for the first time, learning what an application server was, and what a web-application was. Regrettably that team was one of the last to ever develop an end-user application at IBM. Today all IBM software products are huge middleware applications with millions of lines of code developed in teams distributed across the globe among many subsidiary companies and hundreds of programmers. In 2000, he began a job on one of the IBM WebSphere development teams. By then IBM programmers were known as 'developers.' At IBM developers are more "technology wizards" than programmers. One had to be a technology wizard to set up a programming workstation. These tedious tasks involved installing a large stack of software, including the J2EE, the Apache web server, the WebSphere Application Server, DB2, WebSphere Portal Server, and Tivoli Access Manager. This stack of (usually immature) software releases contained many intricate version dependencies which changed about once a month. Learning the version dependencies required expert skill in surfing the IBM internal support websites.

There were few support people for the developers. Many of the UNIX, NT, and WebSphere administrators had been 'out-placed' during the last few annual lay-offs, known internally as the annual "rank and yank" process. As a result, most of the IBM developers did almost no programming at all. They spent their time installing software or testing prerequisite software. However, programming was necessary to display simple looking web pages that interfaced with WebSphere Portal Server. Coding these web pages was anything but simple. It required expert skills in debugging in an n-tier environment. This meant a thorough understanding of J2EE, security, XML, and WebSphere. This was nothing like the functional programming one learned in college. It was pure side-effect programming complete with another overwhelming dose of accidental complexity [2, 7]. Because of the lack of technology support; this kind of work was at best extremely frustrating and at worst impossible. The gratification of design, the steady progress towards results, and the thrill of gaining ultimate control over technology was now completely absent. Instead, one faced non-deterministic diagnostic work in the extreme and because the overall result depended on hundreds of other developers, it was work which gave any individual little control over the final work product. Wright still believes the design of the jobs was very poor [12]. To make matters worse, technical project direction at IBM was usually under the control of non-technical first-line managers. Often these managers were twenty-somethings fresh out of Ivy League MBA programs who lacked a detailed understanding of the technologies involved. These young MBA types relied heavily on technical specialists at IBM called 'architects.'

The architects, few in number, were the IBM technology veterans with at least ten years at IBM; these people were seemingly unbothered by the slow pace of development, the long hours, and the fact the work had almost nothing to do with writing code. The developers depended on these architects to dispense the correct workstation images, the correct test data, the correct method signatures, and the correct build paths. Architects had power and they did not always part with it willingly. Some of the preferred methods of retaining power included dispensing incorrect or incomplete documentation, working off hours to avoid knowledge sharing, and resisting

[4] http://www.keane.com

management efforts to expand the team. After all, it had taken years of heads-down behavior to get to their position as architects, along the way many of their comrades perished in the lay-offs. The architects had learned well the hardest programming lesson that programmers who are dispensable, will be eventually dispensed with. Consequently, IBM architects continued to survive by doing what they had to do to become indispensable; they became experts in UNIX, Linux, DB2, Windows, and WebSphere Portal administration. Architects were highly sought after and highly paid. Regardless, they had little occasion to apply skills learned in school such as correct module decomposition, inheritance lattice construction, list processing, functional programming, etc. In summary, despite the fact Wright loved programming, was good at it, and sought it at every turn his observations pursuing an IT career revealed professional programmers did little programming, and that was before the career was substantially off-shored.

According to the McKinsey Global Institute [11], "a software developer who costs $60 an hour in the U.S. costs only $6 an hour in India." As a result, IBM launched a $200 million development center to handle the bulk of solutions development work for IBM worldwide in 2004 [10]. IBM now has 15% of its global work force in India, or 53,000 of its 356,000 employees. Today the company employs only about 150,000 workers in the United States [5].

Since the mid-1990s a growing array of processes and services formerly conducted in the United States including code writing, software design, and data processing have been outsourced to providers in low wage countries. U.S. imports of business, professional, and technical services associated with off-shore outsourcing rose from $21.2 billion in 1997 to approximately $37.5 billion in 2002, an increase of 77 percent. The most widely cited estimate of the scale of white-collar off-shore outsourcing is a 2002 Forrester projection that "over the next 15 years, 3.3 million U.S. services industry jobs and $136 billion in wages will move off-shore" [9].

For the IT industry off-shoring is no longer simply an experiment with alternative service delivery; it has become an established business practice. There is little doubt the growth of off-shore outsourcing has negatively affected both job availability and wages in the job market which compounds the difficulties faced by unemployed IT workers in what has been a prolonged downturn [17]. According to a recent report released by the Brookings Institution, one of every five programming and software engineering jobs in Silicon Valley will be lost to off-shoring over the next decade.[5] The high salaries in Silicon Valley are cited as a major reason. The study predicted that about 3.4 million of the U.S. jobs will be off-shored to other low cost destinations, such as India and China, between 2000 and 2015. The Brookings Institution also reports about 20 percent of the existing U.S. IT jobs will be lost to such low cost destinations by 2015. This study revealed about 60,000 well paying jobs will be off-shored between 2004 and 2015. Many jobs could disappear from specific metropolitan areas. San Jose is expected to lose 24% to 26% of the programming and software engineering jobs that existed in 2004. This area is forecast to lose almost 43% of the 52,510 jobs that existed three years ago. Other top vulnerable metro areas are San Francisco, CA, Boulder, CO, Lowell, MA, Stamford, CT, and Dallas, TX. In Dallas about 49,000 to 57,000 jobs will be lost to other countries over the next eleven years. Research predicts the average rate of job loss in the United States due to off-shoring will be unusually high in Connecticut, Massachusetts, New Jersey, Colorado, Texas, and California [17].

Besides off-shoring, the IT industry still suffers from the 2001 recession as well as the ensuing jobless recovery [17]. Between March 2001 and November 2001 the IT industry shed 197,000 jobs, or 9.2 percent of employment. Further the industry fared no better during the recovery. By March 2002 (one year after the start of the recession) IT industry employment had declined by more than 270,000 jobs. Significant losses continued such that by March 2003, industry employment had fallen by an additional 113,000 jobs. Although the industry still suffered a decline of 19,700 jobs, employment losses finally slowed by March 2004. All told these mounting losses meant the industry lost about 402,800 jobs between March 2001 and March 2004. Of these lost jobs 205,000 (over 50 percent) were lost during the economic recovery [17].

Astonishingly, industry leaders and economic policy makers repeatedly proclaimed the future of the American workforce is in information technology and knowledge-based jobs working as what Robert Reich termed "symbolic analysts" [15]. Supporting these claims are signs, since March of 2004, of a weak resurgence in the IT

[5] www.brook.edu/metro/pubs/20070131_offshoring.htm

industry. The total number of available positions increased almost 50% since April 2002 if Monster.com is a reasonable indicator [17]. The U.S. Bureau of Labor Statistics forecasts a 20%–50% job growth in all computing specialties by 2012, except for computer operations which is declining and programming which is flat [9]. It is widely reported that finding people to fill the growing number of IT jobs will soon be more difficult than ever. But as we have seen, evidence suggests the vast majority of the U.S. jobs will not be programming oriented and not highly paid. And if something like Senate Bill *S.B. 1348: The Comprehensive Immigration Reform Act of 2007* is passed then the permitted number of H1B visas would more than double and further depress wages for American programmers.[6]

The web has radically changed the demand for programming skills [13]. Web programming which is mainly side-effect programming jumped into a commanding lead in the total number of jobs requiring programming skills and is now mentioned in an impressive 42.6% of programming job ads. Web programming, UNIX, SQL programming, and Oracle database seem to be in the top six skills in the current job market. Each of these skills is demanded in at least 20% of advertised IT jobs. Web programming comprises a complex set of non-functional programming skills including scripting languages and meta-languages such as XML. The demand for UNIX, Linux, and Windows skills seems to be holding steady. Over half of the 2005 job advertisements listed skills in one of these three operating systems as necessary or desirable. Demand for database, ERP (such as SAP, Oracle, and PeopleSoft), and e-commerce servers skills (such as WebSphere and WebLogic) are on the rise [13].

COMPUTING CURRICULUMS IN THE FUTURE

Based on experience it would be unwise to design future computing related curriculums solely on the above employment figures. As you know, the arbitrariness of the evolution of computing standards permits demand for some skills to persist while that for others die quickly. When designing curricula it is often beneficial to use retro-thinking and consider first the characteristics of the capstone courses.

Many capstone courses involve locating real world programming projects in the community [8]. However, this often consumes too much time as professors teach their students during limited semesters. Teachers trudging about town looking for programming projects waste time. Further, real projects are usually well beyond the grasp of most new graduates. We probably do students a tremendous disservice by giving them the impression they are likely to make a living as a professional software writer. It is true there will be those gifted individuals who do make a living as a programmer, but today it is more unlikely. Other capstone computing courses have students working individually on long imaginary projects developed from scratch that gives them a chance to do their own design and build something they feel they own. Unfortunately, this may give students a poor understanding of the real world. The real world is not about working alone. It is about serving others. If students finish the course thinking, "I built something for me; I can profit that way," then academics have probably failed.

In brief, to survive in modern times, many computing related degree programs need a complete make over. The Association for Computing Machinery (ACM) education board's Great Principles of Computing Project seeks to portray the field in terms of its fundamental principles and core practices. Programming is only one of four core practices. The other three are: systems thinking, modeling, and innovating. The authors believe deemphasizing programming and giving systems thinking a more prominent role in computing curricula would go a long way toward improving its effectiveness and appeal. (For an excellent discussion of creative ideas for adding innovation to Computer Science curricula we suggest [4].)

Herbert A. Simon is perhaps the most well-known proponent of systems thinking in the computing sciences.[7]

[6] http://www.washingtonwatch.com/bills/show/110_SN_1348.html.

[7] Simon characterized Artificial Science (as opposed to natural science) as those phenomena whose behavior is contingent upon the purpose of the designers.

> …(computers) can be studied in the abstract, namely using mathematics. Yet, they can and must also be studied empirically….the behavior of computers will turn out to be governed by simple laws, the apparent complexity resulting from that of the environment they are trying to adapt to [16].

Historically, computing curricula have not emphasized the empirical study of these artificial systems. A new field called "Service Sciences" is emerging in academia in which the empirical study of artificial systems will play an undeniable part.

Service Sciences

Service sciences is emerging because the economies of the major industrialized nations are now more than 75% services based. As a result service sciences have been mentioned in recent articles in the Financial Times, Business Week, Harvard Business Review, Technology Review, and The New York Times [14]. Service sciences apply insights from computer science, operations research, engineering, management, social, cognitive, and legal science to analyze how to align people and technology (socio-technical systems) in a more productive way to generate value for both service providers and clients. You cannot discuss services without considering people and ethics both in terms of building lasting relationships and in understanding the dynamics of the socio-technical interface between the provider, client, and the technological processes.

At IBM services are said to have the following unique properties:

- all economic activity whose output is not a physical construct,
- an activity provided as a solution to customer problems,
- a change in state of an economic entity caused by another,
- intangible and perishable,
- created and used simultaneously,
- co-produced by the customer and the provider,
- a time-perishable experience, and
- highly customized.

Thus, a service is a provider/client interaction that creates and adds value. For instance, in a typical doctor/patient interaction both sides benefit from the transaction and it is referred to as "capturing value." The doctor receives a fee, the patient gets a health assessment, and (it is hoped) recovers from the illness. This basic principle also underlies the interaction between any service provider and consumer. The provider and client synergistically coordinate their work (co-production) and in the process both create and capture value. Services typically require an assessment during which the provider and client come to understand one another's goals and capabilities. In the case of the doctor/patient interaction, the patient checks to see if the doctor is licensed and/or accredited and if he or she has the right specialty for the given illness. The doctor also conducts an assessment to determine the patient's current ailment, medical history, and to verify payment details. All these steps factor into both sides capturing value from the service. Obviously for IT and business services these assessments can be far more complex, but the processes and measurements are very similar. Often if the client of a service does not train the necessary service people in a reengineered process for example, the client will not receive the best benefits of the service. The provider in many cases must monitor and assess the way the client is performing their duties and responsibilities. Also, the client needs to determine that the provider is likewise applying satisfactory effort and quality controls in the performance of their tasks. These issues become more important when outsourcing, especially when the client outsources a component of its IT business to a provider that is in a different country with different government regulations and national cultures.

Service sciences have had a low priority in academe until recently. However, we believe the empirical study of artificial systems will become an increasingly important part of what is considered to be service sciences and a primary paradigm by which future computing related curriculum are designed. For example in 2004, IBM

Research published the white paper *"Services Science: A New Academic Discipline*?" [8] Presently IBM Research is working with Arizona State**,** Carnegie Mellon, and Penn State universities on joint research projects. In 2006, North Carolina State University became the first American research university to offer new master's-level curricula in Service Sciences in its Colleges of Engineering and Management [1].

PROPOSED ARTIFICIAL SYSTEMS AND SERVICE SCIENCES PROGRAM

Surveys of high school students reveal they are well aware of the diminishing career prospects in computing related disciplines. Research shows students find other technical fields such as bioinformatics and molecular biology more attractive [6]. So why not look to the medical school curriculums for some guidance for designing curricula for an Artificial Systems and Service Sciences degree program?

For example, at Yale Medical School the first year consists of basic sciences in normal body structure, functions, growth, and development. In the second year more emphasis is placed on disease and its treatment including pathology, pharmacology, microbiology, immunology, and how to take a medical history. During the third year students receive clinical training by rotating through various hospitals and ambulatory services. All students participate in the following seven clerkships:

1. Internal Medicine - This deals with non-surgical treatment of diseases in adults, especially of internal organs.
2. Surgery
3. Pediatrics – Deals with the medical care of people 0-21 years.
4. Neurology
5. Family Practice - Deals with treatment of acute and chronic illnesses, and provides preventive care and health education for all ages and both sexes. Some also care for hospitalized patients, do minor surgery and/or obstetrics.
6. Obstetrics and Gynecology - These are surgical specialties dealing with the female reproductive organs.
7. Psychiatry [18].

The fourth medical school year is an elective year with 232 electives offered. Each elective is four weeks long. Where standard elective choices seem too limiting, a student is encouraged to approach individual faculty members to develop courses that more closely approach their individual needs. The year is divided into nine four-week periods.[9]

The authors believe that service sciences have much to learn from medical school curriculums. In that spirit, the following is an adaptation of the Yale Medical School curriculum to what we call the School of Artificial Systems and Service Sciences, which would educate and train management practitioners. Thus, we propose the following:

Mission Statement

To holistically educate practitioner-scholars and future leaders who will advance the management practice of artificial systems and service sciences.

Knowledge Objectives

Students will acquire and demonstrate a basic understanding, theoretical as well as empirical, of the physical, psychological, economic, and cultural matters that affect the management of artificial systems and service sciences. Knowledge consists not only of information but also includes the critical understanding of how that information is obtained, expanded, and renewed. One's knowledge base must include the ability to augment one's self through a lifetime of learning and scholarship. Students must couple this self-renewing capacity with an ability

[8] http://domino.research.ibm.com/comm/www_fs.nsf/images/fsr/$FILE/summit_report.pdf
[9] http://www.med.yale.edu/education/edu/mission.html

to evaluate new information critically and contribute to the discovery of new knowledge by engaging in an independent research project culminating in a senior thesis. The school would ensure that graduating students demonstrate, to the satisfaction of the faculty, the following knowledge, skills, and abilities:

- An understanding of the typical structure of artificial systems at the holistic, service, process, processor, memory management, and micro instruction level.
- An understanding of best practice functioning at the holistic, service, process, processor, memory management, and micro instruction level.
- Understanding of the normal anatomic and physiologic changes that occur over the lifecycle of an artificial system.
- An understanding of the physics of electronic computers.
- An understanding of the psychology of the perception of how users, developers, administrators, and managers view artificial systems during their lifecycle.
- An understanding of the etiology of artificial system defects including incorrect requirements and design, maintenance and release interval errors, malware, and tool defects.
- An understanding of basic epidemiologic principles and the use of statistics in describing defects within defined system populations.
- An understanding of the major challenges, both present and future, to the electronic health of communities and nations, as well as a familiarity with the prevention and treatment strategies needed to address these challenges.
- Knowledge of the clinical, laboratory, diagnostic imaging and pathologic manifestations of system defects, and proficiency in interpreting each type of information. Depth of understanding should be sufficient to allow for creation of appropriate differential diagnoses and establishment of additional investigative strategies, when needed.
- A critical understanding of the principles of both curative and palliative therapeutics. This includes objectives of repairs, assessment of efficacy and risks, and awareness of the common use of alternative and complementary repairs.
- Knowledge of the strategies needed to prevent defects in artificial systems.
- Knowledge of existing systems service capabilities in the United States and other countries, and familiarity with systems service management strategies to develop systems (both large scale and small) that maximize systems service effectiveness, reduce errors, efficiently utilize scarce resources, and to correct inequities in access to systems service.
- Awareness of the necessity for scientific method and knowledge of its application to force the discovery of new knowledge [18].

Skills Objectives

The school would educate management practitioners who demonstrate skills in the core set of activities required for artificial systems and service sciences. At the heart of these diagnostic activities are history taking and physical examination which would be complemented by facility evaluations that focus on the appropriate use and interpretation of procedures and tests. Students must demonstrate the ability to acquire, interpret, and apply information from diverse sources to develop their diagnostic and management skills. Additionally, students learn to communicate more effectively while carrying out these activities.

The school would ensure graduating students demonstrate, to the satisfaction of the faculty, the following:

- The ability to obtain an accurate systems maintenance history that covers relevant and essential aspects of that history, including issues related to age, manufacturers, and maintenance budget while recognizing and addressing any barriers to communication.
- The ability to perform both a complete and focused clinical examination in a manner that demonstrates respect for systems stakeholders.
- An understanding of the clinical method: the process whereby information obtained from the history, physical examination, and laboratory data is formulated into a differential diagnosis.

- The ability to formulate a plan of service that takes into consideration the system stakeholders, goals of service, the risks, benefits, alternatives, and financial consequences of each option.
- The ability to find, identify, critically interpret, and utilize the relevant information from both printed and electronic media and apply the scientific method in order to practice informed, up-to-date systems service.
- An understanding of the principles of artificial systems scholarship, including literature review, design of hypothesis, formulation of specific aims, identification and application of state-of-the-art methods including statistics, analysis, and interpretation of data which includes familiarity with the elements of clinical and translational research and knowledge of responsible ethical research conduct.
- The ability to communicate effectively with systems stakeholders, colleagues, and others with whom the management practitioner must exchange information in carrying out their responsibilities. These abilities should include proficiency in the education of systems stakeholders, inquiries about management control strategy, operator error, breaking bad news, counseling for behavior modification, obtaining informed consent, and discussions of end-of-life issues.
- Knowledge of the indications for a core set of systems service procedures, as well as the complications of those procedures. Students must demonstrate ability to obtain consent for the procedure, perform the procedure, and to recognize and interpret the results.
- The ability to develop a scientific question, survey the literature, design and carry out a study to address that question, and prepare a written presentation of that work in the form of an independent strategic research project.
- The skills required to be a life-long management practitioner-scholar, including the ability to assess the validity of the literature and to objectively apply the scientific method to problem-solving and decision-making [18].

Student Attitudes

Artificial systems and service sciences students should strive to be more altruistic. Practitioners must be committed to serving others and devoted to the care of their clients. They must bring ethical intent and action, as well as empathy and compassion, to their physician-client type of relationship. Managers must demonstrate honesty and integrity in all of their professional interactions. Empathy requires curiosity and a willingness to feel, perceive, and understand the experiences of others. These management practitioners must listen to their "patient's story" to understand the client's experience of the system malfunction in the context of their beliefs, values, personal circumstances, and unique human qualities, and respond compassionately based on the clients' concerns. Management practitioners have the responsibility to be aware of their own reactions and emotions with special attention on how this influences attitudes toward and behavior with their clients. The School of Artificial Systems and Service Sciences would admit students who demonstrate humanitarian values, maturity, and the capacity for self-reflection. Through its curriculum and educational approach, the school would provide opportunities for students to maintain, preserve, and enhance the empathy and compassion that attracted them to artificial systems and service sciences so it continues to be evident in their development.

The school would ensure graduating students demonstrate, to the satisfaction of the faculty, the following:

- Empathic care of systems stakeholders through their interest in how they experience and cope with system defects.
- Respect for stakeholder dignity, including the right to privacy and confidentiality.
- Knowledge of the principles that guide ethical decision-making and awareness of the major ethical dilemmas in service sciences, including those arising at the beginning and end of life, those posed by the expansion of science and technology, and those resulting from financial constraints and incentives.
- Honesty, accuracy, and integrity in all interactions with systems stakeholders and colleagues, including scientific integrity.
- The ability to recognize and accept the limitations in one's knowledge and skills, along with an ongoing, lifelong commitment to improve one's knowledge and abilities as a service sciences practitioner.
- An ethical commitment to understand and advocate for systems stakeholders interests and those of the community over one's own personal interests.

- An awareness of one's vulnerability to stress and the influence stress has on the ability to care for one's clients.
- An awareness of the possibility of a management practitioner's bias, arising both from personal background, conflicts of interest, the culture of artificial science, and the ways that bias can affect the delivery of systems care and the physician-client type relationship [18].

Context Of Artificial Systems And Service Sciences

Neither clients nor IT management practitioners exist in isolation. Students must understand the context in which the clients live and the complex interactions between heterogeneous systems and the environment. Students realize they will practice their profession within a context and must understand the needs and expectations of society at large as well as have the ability to work with colleagues. Further, students should realize being an IT management practitioner is a privilege - one that comes with a responsibility to serve the community that has empowered them by providing access to their training and knowledge.

Thus the school would also ensure graduating students demonstrate, to the satisfaction of the faculty, the following:

- An ability to understand the multi-factorial nature of artificial systems and relate such insight to the context of the individual client, including, but not limited to, psychological, social, economic, and cultural factors.
- The ability to elicit the client's goals, values, and preferences in an understanding manner and to integrate these values and preferences into an appropriate plan of service.
- An awareness of one's participation as a member of a service team and the ability to collaborate appropriately with all those involved while respecting their roles and professional expertise.
- Demonstrate a theoretical and practical understanding of how artificial systems service is delivered, and how the manner of that delivery can affect the individual practitioner.
- An understanding of the utility and limitations of evidence-based decision-making and cost-effective service delivery.
- An appreciation of the artificial systems service profession's responsibility to society, both in our own country and throughout the world. These responsibilities should include not only service to the underserved or disenfranchised members of our own society, but also advocacy for the care of the disadvantaged persons in other nations [18].

CONCLUSION

In conclusion, we believe at this juncture in the history of artificial systems and service sciences there are many lessons to be learned from the behavioral sciences and medicine. Students training in artificial systems and service sciences need to have more breadth and depth in critical areas. For example, in medical schools the entire third and fourth years consist mainly of internships in the various major specialties. What are the appropriate major specialties of artificial systems and service sciences? Industry best practice frameworks for service management, such as the IT Infrastructure Library (ITIL) certainly provide guidance.[10]

The Internal Medicine internship for physicians becomes the Service Desk internship for IT management practitioners. Here students would serve as assistants to systems support specialists at large data center service desks. Students would handle routine first line support, especially incident recording and escalation. Surgery rotation becomes Change Management rotation for IT management practitioners. Students work with programmers and change control specialists. Program maintenance is stressed. The Pediatric rotation becomes the *Programming* internship. In this rotation, students work on programming teams developing new systems from scratch. Requirements analysis, tool usage, version control, and system design would be strongly emphasized. The Neurology internship becomes the *networking* technology rotation where students learn and work with network specialists. Analogous to the Family Practice medical specialty would be *Problem Control*. In this module, students work with system administrators. They learn what UNIX and Windows administrators really do. Finally and

[10] http://www.itil.co.uk

perhaps most importantly, analogous to the Psychiatry rotation could be what ITIL calls *Service Level Management.* This interdisciplinary approach is required to understand how services are conceived, designed, delivered, and supported. Students work as executive assistants to first line IT managers. Emphasis is on the writing of service level agreements, underpinning contracts, and operational level agreements.

Examining the analogies between artificial systems and service sciences and medicine is a fascinating area for research. Recently a workshop was hosted by the Information Technology Service Management Forum to promote awareness about opportunities for integrating service sciences into curricula with artificial systems with service sciences scholars having an opportunity to play a major role in leading the cross-disciplinary curricula design.[11] Realizing and understanding this potential requires investigation of technology, human behavior, strategy, design, and economics.

As the complexity of IT systems increase, the analogy of artificial systems and service sciences curricula to those of medical schools, and the lessons thereby learned, will undoubtedly become stronger. Perhaps the most important lesson will be that artificial systems and service sciences students be trained to be both first-rate diagnosticians and ethical problem-solving management practitioners as well.

REFERENCES

1. Allen, S.G., Mugge P. & Wolff, M.F. Services Science To Be Taught At NC State. *Research Technology Management*, Nov/Dec. 2006, Vol. 49 Issue 6.
2. Brooks, Frederick P. No Silver Bullet: Essence and Accidents of Software Engineering. *Computer*, Vol. 20, No. 4 (April 1987).
3. Dijkstra, E.W.G. *Go To Statement Considered Harmful. Communications of the ACM 11*, 3 (Mar.1968).
4. Denning P. J. and McGettrick. Re-centering Computer Science. *Communications of the ACM* November 2005/Vol. 48, No. 11 15.
5. Engardio, B & Kripalani. IBM May Rival India's Top Offshore IT Firms in Head Count. *Computerworld*. April 04, 2007.
6. Foster, A. Student Interest in Computer Science Plummets. *Chronicle of Higher Education* 51, 38 (May 7, 2005), A31.8.
7. Hudak P. Conception, Evolution and Application of Functional Programming Languages. *ACM Computing Surveys*, Vol. 21, No. 3, September 1989
8. Martin, F. Toy Projects Considered Harmful. *Communications of the ACM*. July 2006/Vol. 49, No. 7.
9. McCarthy, John C. 3.3 Million U.S. Services Jobs to go Offshore. *Forrester*, November 11. http://www.forrester.com/ER/Research/Brief/Excerpt/0,1317,15900,FF.html. 2002.
10. McDougall, P.C. IBM To Move All Solutions Development Operations To India. *InformationWeek* March 8, 2006.
11. McKinsey Global Institute. *Off-shoring: Is it a Win-win Game?* San Francisco: McKinsey & Company. 2003.
12. Parker S, & Wall T.D. *Job and Work Design: Organizing Work to Promote Well-being and Effectiveness*. Sage Publications, ©1998.
13. Prabhakar ,B. & Litecky, C. R. IT Skills in a Tough Job Market. *Communications of the ACM* October 2005/Vol. 48, No. 10 91.
14. Rai A. & Sambamurthy V. (2006). Editorial Notes – The Growth of Interest in Services Management: Opportunities for Information Systems Scholars. *Information Systems Research*. No.4, December 2006, pp. 327-331.
15. Reich, R. *The Work of Nations*. New York: Alfred A. Knopf. 1991.
16. Simon, H. A. *The Sciences of the Artificial - 3rd Edition*. The MIT Press; 3 edition. October 1, 1996.
17. Srivastav S and Theodore, N. (2005). A Long Jobless Recovery: Information Technology Labor Markets after the Bursting of the High-Tech Bubble. *The Journal of Labor and Society* · 1089-7011 · Volume 8 · March 2005.
18. www.med.yale.edu/education/edu/mission.html

[11] http://www.itsmf.org

NOTES