Software and System Engineering Education: Commonalities and Differences

Dr. Massood Towhidnejad, Embry-Riddle Aeronautical Univ., Daytona Beach

Massood Towhidnejad is the director of NExtGeneration Applied Research Laboratory (NEAR), and a tenure full professor of software engineering in the department of Electrical, Computer, Software and System Engineering at Embry-Riddle Aeronautical University. His research and teaching interests include autonomous systems, and software and systems engineering with emphasis on software quality assurance and testing.

Dr. Thomas B Hilburn, Embry-Riddle Aeronautical Univ., Daytona Beach

Dr. Thomas B. Hilburn is a Professor Emeritus of Software Engineering at Embry-Riddle Aeronautical University and was a Visiting Scientist at the Software Engineering Institute, Carnegie-Mellon from 1997 – 2009. He has worked on software engineering development, research, and education projects with the FAA, General Electric, Lockheed-Martin, the Harris Corp, the MITRE Corporation, DOD, FIPSE, the SEI, the NSF, the ACM and the IEEE Computer Society. His interests include software processes, object-oriented analysis and design, formal specification techniques, and curriculum development, and he has published over 70 papers in these areas. He is an IEEE Certified Software Developer, an IEEE Software Engineering Certified Instructor, and has chaired committees on the Professional Activities Board and the Educational Activities Board of the IEEE Computer Society.

Dr. Richard Fairley, Software and Systems Engineering Associates (S2EA)

Richard E. (Dick) Fairley, Ph.D., CSDP is principal associate of Software and Systems Engineering Associates (S2EA) and an adjunct professor in the graduate software engineering program at Colorado Technical University in Colorado Springs. He is chair of the IEEE Computer Society's Software and Systems Engineering Committee and chair of the Computer Society's Special Technical Committee on Systems Engineering. In addition, he is a member of the BKCASE governing board and is the Computer Society delegate to INCOSE. He is also chair of a joint Computer Society – Project Management Institute committee that is developing a software project extension to the PMI Guide to the Project Management Body of Knowledge and is the Computer Society delegate to PMI. Dr. Fairley In his career of 30+ years, Dr. Fairley has been a tenured professor, department chair, academic dean, and trainer and consultant. He holds bachelors and masters degrees in electrical engineering and a Ph.D. in computer science and applied math. Dr. Fairley is a member of IEEE, the IEEE Computer Society, INCOSE, and PMI.

Software and Systems Engineering Education: Commonalities and Differences

Abstract

The complexity of current engineered systems and the increasing role of software in those systems emphasize the need for educating and training qualified systems engineers to meet future demand. As the role of software increases in the overall operation and success of systems it becomes necessary for the system engineers to understand and appreciate the software engineering methods and practices. The same is true for software engineers, where they need to understand the overall complexity of the systems they deal with, and understand systems engineering methods and practices. Over the past five years, educators and professional software and systems engineers from around the world have been working on the development of two graduate reference curricula: the Graduate Software Engineering Reference Curriculum and the Graduate Reference Curriculum for Systems Engineering. This paper provides an overview of these curricula, and how they might influence the quality and effectiveness of the development of future Software Intensive Systems. In addition, the paper presents some of the commonalities and differences between software engineering and system engineering graduate education, and a discussion of the challenges involved in graduating engineers, qualified for work on software intensive systems.

Introduction

In the last twenty years there has been much effort devoted to enhancing and advancing the state of professional software engineering (SwE) and systems engineering (SE) practice. This effort has been driven by two issues: (1) software and systems engineering are relative new fields of engineering, considered by many not to have reached the maturity of more conventional fields of engineering; (2) the complexity of engineered systems has increased dramatically in the past fifty years, with the role of software playing an increasing critical element in such systems, so called Software Intensive Systems (SISs). These issues highlight the need for educating and training qualified SIS engineers to meet current and future demand.

Over the past five years, educators and professional software and systems engineers from around the world have been working on the development of two graduate reference curricula: the Graduate Software Engineering Reference curriculum (GSwE2009^{TM}), and Graduate Reference Curriculum for Systems Engineering (GRCSE^{TM}). These works have been supported by a number of international professional organizations such as the ACM, the IEEE Computer Society, the IEEE Systems Council, and INCOSE. This paper provides an overview of both the GSwE and GRCSE documents, and how they might influence the quality and effectiveness of the development of future Software Intensive Systems. In addition, the paper presents some of the commonalities and differences between software engineering and system engineering graduate education and a discussion of challenges involved in educating engineers, who arequalified for work on software intensive systems.

Graduate Software Engineering Reference Curriculum

The GSwE2009 project team, consisting of forty-three authors from more than 24 organizations, was formed in the summer of 2007 and worked for two years developing the *Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering*[1, 2]. An underlying focus of GSwE2009 was how to advance the state of software engineering practice and to support a better understanding and agreement about the nature of "professional software engineers". The full GSwE2009 document, with more detailed information, can be downloaded at www.gswe2009.org.

The key features of GSwE2009 are the following:

- A set of guiding principles that set forth the fundamental philosophy for GSwE2009 development.
- A set of student outcomes that guide the development of a curriculum by setting out the general capabilities of graduates.
- A baseline set of student skills, knowledge, and experience assumed by the core curriculum, with provisions for individual institutions to reset these, provided their programs achieve the desired outcomes.
- A description of the fundamental core skills, knowledge, and practices to be taught in the curriculum to achieve the outcomes, termed the GSwE2009 Core Body of Knowledge (CBOK).
- An architectural framework that supports a flexible curriculum implementation by allowing each university to fashion a program guided by its own specialties and culture.

GSwE2009 Curriculum Architecture

The student outcomes guided and controlled the development of both the structure and content of the GSwE2009 curriculum. The structure of the GSwE2009 curriculum is represented in the architectural model depicted in Figure 1. It identifies, via the CBOK, the minimal material that all programs should include and makes provisions for each institution to develop its own distinctive program(s). The curriculum architecture is compatible with existing master's programs, for which course and curriculum data are described in[3]. It is intended to provide a structural basis for programs based on the GSwE2009 outcomes.

The curriculum architecture includes preparatory material, core materials, university-specific materials, elective materials, and a capstone experience. The heavy black line in Figure 1 represents the baseline preparatory knowledge for students in a GSwE2009 master's program. For example, the preparation might be achieved through an undergraduate computing or engineering degree, plus two years of software development experience. However, the preparatory knowledge is not meant to represent admissions requirements, but rather material that a program must ensure students acquire in order achieve the prescribe outcomes. Material below the heavy black line is mastered after the baseline preparation is achieved.

GSwE2009 strongly recommends that students demonstrate their accumulated skills and knowledge in a capstone experience, which might be a project, a practicum, or a thesis. Students completing the curriculum must be able to understand and appreciate the importance of teamwork, negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

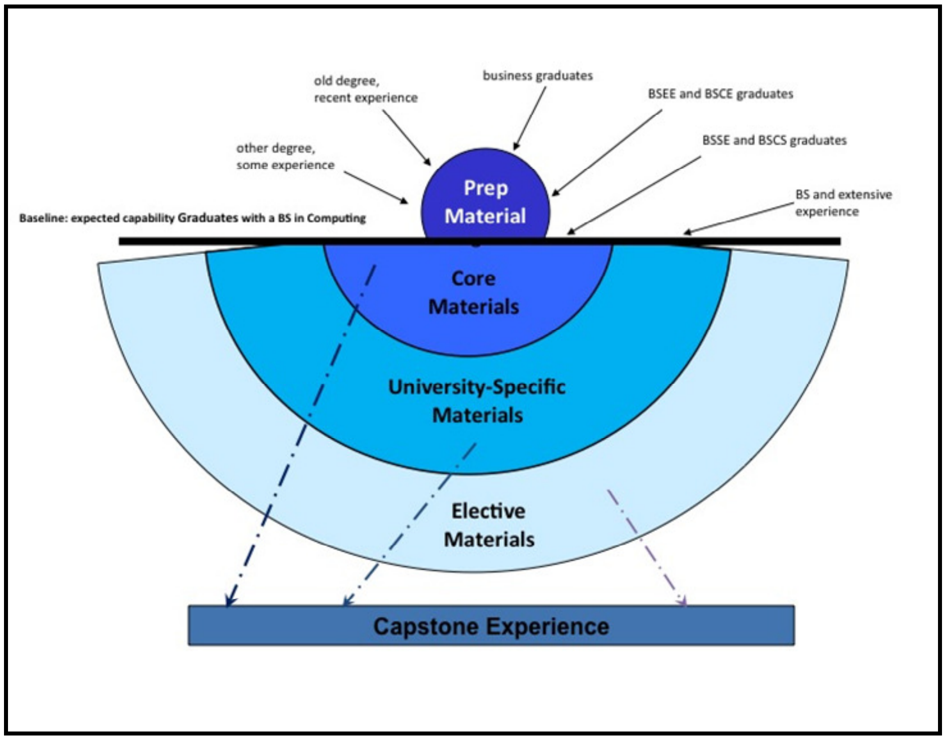
old degree, recent experience
business graduates
BSEE and BSCE graduates
other degree, some experience
BSSE and BSCS graduates
Prep Material
BS and extensive experience
Baseline: expected capability Graduates with a BS in Computing
Core Materials
University-Specific Materials
Elective Materials
Capstone Experience

Figure 1: Architectural Structure of a GSwE2009 Master's Program

GSwE2009 Core Body of Knowledge

The GSwE2009 curriculum content consists primarily of the CBOK and its extensions, which are strongly related to the CBOK, DOMAIN, DEPTH, SYS ENG, ETHICS, and RECONCILE outcomes (see Table 3). Figure 2 depicts the organization of the Core BOK, with percentage of curriculum content designated for each core area. Notice that the CBOK occupies approximately 50% of the curriculum allowing flexibility and specialization in curriculum design and supporting its extension to support outcomes DOMAIN and DEPTH.

The primary source for developing the CBOK was the SWEBOK[4.] Knowledge elements were also derived from SE2004[5] and other sources[6, 7, 8]. In the study and analysis of these sources, it was decided that although the SWEBOK organization and content would dominate, various changes in areas and topics were needed to support the GSwE2009 expected student outcomes and to accommodate the needs and views of academia, industry, and the computing professional societies. For example, two knowledge areas, not in the current version of the SWEBOK, were added: Systems Engineering Fundamentals, and Ethics and Professional Conduct.

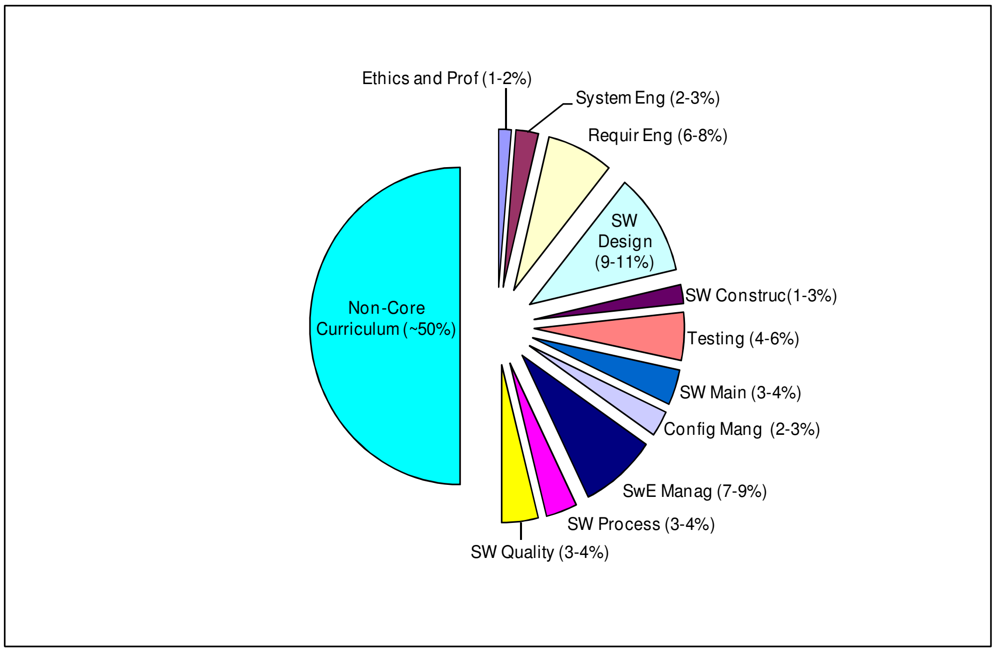

Figure 2: CBOK Organization

Graduate Reference Curriculum for Systems Engineering

The BKCASE (Body of Knowledge and Curriculum to Advance Systems Engineering) project [9, 10, 11] is three-year project, begun in September 2009 and sponsored by the Department of Defense, to develop a guide to the Systems Engineering Body of Knowledge (SEBoK) and a Graduate Reference Curriculum for Systems Engineering (GRCSE). Additonal information about BKCASE and a link to GRCSE is available at http://www.bkcase.org. The GRCSE project team, part of the larger BKCASE team, consisted of 19 authors from 14 international organizations.

The purpose of GRCSE is to assist in the development of new master's SE programs and to improve existing SE graduate programs. The principal GRCSE stakeholders are universities, students, graduates, employers, and systems customers and users. GRCSE is designed to support a systems-centric program and a professional master's degree focused on developing student ability to perform systems engineering tasks and roles. The key features of GRCSE are similar to those discussed for GSWE2009.

GRCSE Curriculum Architecture

The GRCSE curriculum architecture organizes the topic areas that address the knowledge, skills, and abilities a student should learn in order to achieve the expected outcomes upon graduation with a master's degree in SE. The architecture does not organize the courses within which the topics will be included in a particular graduate program.

Figure 3 provides a visualization of the architecture. The figure demonstrates how a student might progress through the curriculum components: starting with satisfying the basic entrance expectations for education (e.g., in engineering, mathematics, computing, and communication) and experience (realistic SE practice experience); then engaging in educational activities related to the CorBoK Foundation and Concentration areas; and eventually involvement in a capstone experience, in which students can demonstrate their ability to bind together all aspects of the program.

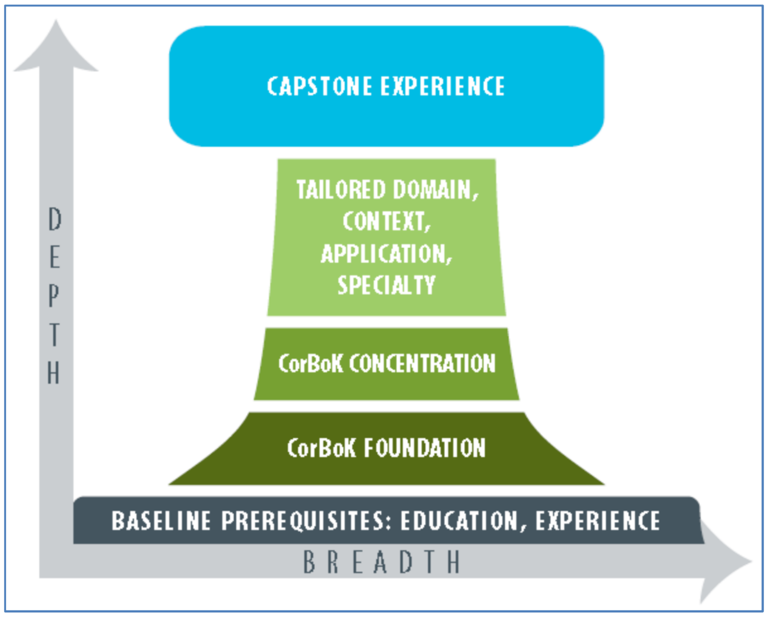
CAPSTONE EXPERIENCE
TAILORED DOMAIN, CONTEXT, APPLICATION, SPECIALTY
CorBoK CONCENTRATION
CorBoK FOUNDATION
BASELINE PREREQUISITES: EDUCATION, EXPERIENCE
DEPTH
BREADTH

Figure 3: GRCSE Curriculum Architecture

GRCSE Core Body of Knowledge

The CorBoK is based on the SEBoK. The SEBoK contains over 100 SE topics and is organized into the seven parts, as described in Table 1. The SEBoK can be viewed online at http://www.sebokwiki.org/1.0.1/.

Table 2: SEBoK Organization

SEBoK Part	Part Content
Part 1: SEBoK Introduction	Covers the scope, structure, uses, and evolution of the SEBoK.
Part 2: Systems	Describes the characteristics of systems and foundation principles of SE.
Part 3: SE and Management	Addresses how SE is conducted and covers life cycle models and processes, SE development and evolution practices, management processes, and standards.
Part 4: Applications of SE	Covers the application of SE to the development and deployment of products, services, enterprises, and systems of systems.
Part 5: Enabling SE	Discusses the enabling of SE at the individual, team, and business/enterprise levels and includes a discussion of ethics, team dynamics, and culture.
Part 6: Related Disciplines	Focuses on the relationship of SE to other disciplines.
Part 7: SE Implementation Examples	Includes overviews of case studies and vignettes, which provide real-world examples of SE activities and provide links back to the concepts covered in the first six parts of the SEBoK.

The CorBoK topics are those from Parts 2 through 6 and are structured in two parts: foundation knowledge and concentration knowledge. Foundation knowledge is composed of the broad set of topics deemed essential for all systems engineers. Each student is also expected to choose and acquire in-depth knowledge in an area of concentration. GRCSE includes two example concentrations: Systems Engineering Management (SEM) and Systems Design and Development (SDD).

The CorBoK was developed so that it comprises approximately 50% of the curriculum. For purposes of planning curriculum content, GRCSE estimates 480 hours as the total number of contact hours needed for a SE master's program. Consequently, using the 50% guideline, CorBoK instruction is presumed to take approximately 240 contact hours. Table 2 provides an example:

- Approximate contact hours and percentages of the 50% devoted to CorBoK are provided for each part.
- Since Part 3 represents a significant portion of the 50%, distribution by knowledge area is presented.
- The example is meant to be typical, but many other distributions are clearly possible.
- The example distribution was determined by engaging a group of GRCSE authors in a quasi-Wideband Delphi technique to allocate the 240 contact hours.

Table2: Example Distribution of Time for the CorBoK

CorBok Part	Foundation/SEM Contact Hours (%)	Foundation/SDD Contact Hours (%)
Part 2: Systems	29 (6%)	29 (6%)
Part 3: Systems Engineering and Management	134 (28%)	134 (28%)
Life Cycle Models	19 (4%)	19 (4%)
System Definition	19 (4%)	29 (6%)
System Realization	19 (4%)	29 (6%)
System Deployment and Use	19 (4%)	19 (4%)
Systems Engineering Management	24 (5%)	10 (2%)
Product and Service Life Management	24 (5%)	19 (4%)
Systems Engineering Standards	10 (2%)	10 (2%)
Part 4: Applications of Systems Engineering	24 (5%)	24 (5%)
Part 5: Enabling Systems Engineering	24(5%)	10 (2%)
Part 6: Related Disciplines	29 (6%)	43 (9%)
Total Distribution - Contact Hours (%)	240 (50%)	240 (50%)

Meeting the SIS Education Challenge

The two reference curricula, GwE2009 and GRCSE, have number of common features. They are both organized around and focused on a core body of knowledge. The architectures for both curriculum models are similarly organized: expected preparation background, core material, specialty areas, and a capstone experience. The core bodies of knowledge for both curriculum models overlap significantly, in the areas of requirements engineering, project management, architectural design, verification and validation. In addition, software and systems engineering share large number of concepts, methods and techniques, this is evident in the IEEE standard on *System Life Cycle Processes*[12], describing a set of process that overlap between systems and software engineering. Finally, there is a large overlap between the program outcomes for graduate programs in software engineering and systems engineering as it is evident in table 3.

Table 3. GSwE and GRCSE Program Outcomes

GSwE Outcomes	GRCSE Outcomes
CBOK - Master the Core Body of Knowledge	**FOUNDATION** – Achieve the designated Bloom's levels of attainment for each topic contained within the Core Body of Knowledge (CorBoK) foundation. **CONCENTRATION** – Achieve designated Bloom's levels of attainment for each topic contained within one of the CorBoK concentrations, as appropriate for the type of master's program or for an individual student's interest.
DEPTH - Master at least one Knowledge Area or sub-area from the CBOK to the Bloom Synthesis level [13].	**TOPIC DEPTH** – Achieve a Bloom's synthesis level of attainment for at least one topic from the CorBoK (either foundation or concentration).
DOMAIN - Master software engineering in one application domain, such as finance, medical, transportation, or telecommunications, and in one application type, such as real-time, embedded, safety-critical, or highly distributed systems.	**APPLICATION DOMAIN** – Demonstrate the ability to perform SE activities in one application domain, such as defense, aerospace, finance, medical, transportation, or telecommunications. **SPECIALTY** – Apply SE principles in order to address a specialty, such as: security, agility, affordability, or safety-critical or embedded systems. **RELATED DISCIPLINES** – Comprehend the relationships between SE and other disciplines, such as project management, human factors, and other engineering fields, as discussed in the SEBoK, and be able to articulate the value proposition of these disciplines for SE.
RECONCILE- Be able to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, risk, existing systems, and organizations.	**REQUIREMENT RECONCILIATION** – Master the quantitative skills to reconcile conflicting requirements, finding acceptable compromises within limitations of cost, time, knowledge, risk, existing systems, and organizations.
TEAM - Be an effective member of a team, including teams that are multinational and geographically distributed, effectively communicate both orally and in writing, and lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.	**TEAMWORK** – Perform as an effective member of a multi-disciplinary team, effectively communicate both orally and in writing, lead in one area of system development, such as project management, requirements analysis, architecture, construction, or quality assurance, and display leadership capabilities within a team.
ETHICS - Be able to make ethical professional decisions and practice ethical professional behavior.	**ETHICS** – Demonstrate knowledge of professional ethics and the application of professional ethics in decision-making and SE practice.
LEARN - Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.	**PROFESSIONAL DEVELOPMENT** – Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.
SYSTEMS ENGINEERING - Understand the relationship between software engineering and systems engineering and be able to apply systems engineering principles and practices in the engineering of software.	**SOFTWARE IN SYSTEMS** – Demonstrate an understanding and appreciation of the level of software engineering necessary to develop current and future products, services, and enterprise systems.
PERSPECTIVE - Understand and appreciate feasibility analysis, negotiation, and good communications with stakeholders in a typical software development environment, and perform those tasks well; have effective work habits and be a leader. **TECH** - Be able to analyze a current significant software technology, articulate its strengths and weaknesses, compare it to alternative technologies, and specify and promote improvements or extensions to that technology.	
	PROBLEM/SOLUTION EVALUATION – Master the quantitative skills to evaluate alternative system solution strategies, including how well different solutions relate to the identified problem, and express the relevant criteria to ensure solutions are selected against a holistic systems perspective. **REALISM** – Comprehend and appreciate the challenges of applying SE to realistic problems throughout the system life cycle.

In addition to the commonalities between the two programs program outcomes, they both place some emphasis on software intensive systems: the GSwE2009 goal SYS ENG specifies that graduates need to be able to "apply systems engineering principles and practices in the engineering of software"; and the GRCSE goal SOFTWARE IN SYSTEMS calls for graduates to understand "the level of software engineering necessary to develop current and future products, services, and enterprise systems". Many would view software engineering as a specialty of systems engineering, and most would agree, because of the prevalence of SISs, that systems engineers need to be knowledgeable about software engineering techniques. However, current curricula in systems and software engineering do not demonstrate how the two disciplines intersect and depend on each other: many systems engineering programs contain little or nothing about software engineering and most software engineering curricula provide only tangential notice of systems engineering. This represents a significant challenge to effective engineering of software insensitive systems. Developers of such systems, both system engineers and software engineers, need to have knowledge and capability in disciplines. GSwE2009 and GRCSE provide the support for the preparation of these engineers and have potential to transform the education of SIS engineers and significantly enhance the effectiveness and quality of SIS engineering.

References

1. Pyster, A. (Ed.), *Graduate Software Engineering 2009 (GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering*, Integrated Software & Systems Engineering Curriculum Project, Stevens Institute of Technology, September 30, 2009. www.gswe2009.org
2. Ardis, M., Bourque, P., Hilburn, T. Lasfer, K. Lucero, S., McDonald, J., Pyster, A. and Shaw, M., Advancing Software Engineering Professional Education, *IEEE Software*, vol. 23, no. 6, pp. 58-63, July/August 2011.
3. Pyster, A., Lasfer, K., Turner, R., Bernstein, L., and Henry, D., Master's Degrees in Software Engineering: An Analysis of 28 University Programs, *IEEE Software*, vol. 26 , no. 5, pp. 94-10, September/October 2009.
4. Abran, A., Moore, J.W., P. Bourque, P. and Dupuis, R. (Eds), *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, IEEE Computer Society Press and also published as ISO Technical Report 19759:2005, 2004. www.swebok.org
5. ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices, *Software Engineering Code of Ethics and Professional Practice*, Version 5.2, 1999. http://www.acm.org/serving/se/code.htm
6. Haskins, C. (Ed.), *INCOSE Systems Engineering Handbook*, Version 3.1, INCOSE-TP-2003-002-03.1, August 2007.
7. Shaw, M. (Ed.), *Software Engineering for the 21st Century: A Basis for Rethinking the Curriculum,* Technical Report CMU-ISRI-05-108, Carnegie Mellon University Institute for Software Research, March 2005.
8. Tockey, S., *Return on Software: Maximizing the Return on Your Software Investmen*t (1st edition). Addison-Wesley, 2004.
9. Hilburn, T.B., Squires, A., and Madachy, R., A Model for Educating Systems Engineers, *Proceedings of 2012 IEEE International Systems Conference* (SysCon), March 2012.

10. Towhidnejad, M., Ferris, T., Squires, A., Madachy, R., Enabling Systems Engineering Program Outcomes via Systems Engineering Body of Knowledge, *Proceedings of 2013 Conference on Systems Engineering research (CSER),* March 2013.
11. Towhidnejad, M., Hilburn, T.B., An Overview of GRCSE: Graduate Reference Curriculum for Systems Engineering, *Proceedings of 2013 World Congress on Engineering Education (WCEE),* January 2013.
12. IEEE Computer Society, *IEEE Standard for Systems and Software Engineering —System Life Cycle Processes*, IEEE Std 15288-2008, 2008.
13. Bloom, B.S. (Ed.), *Taxonomy of educational objectives: The classification of educational goals: Handbook I*, cognitive domain, Longmans, 1956.