REPORT FROM THE TRENCHES: APPLYING THE SEEK TO BSSE PROGRAM DEVELOPMENT

Stephen Frezza[1], Sreela Sasi[2], Jaehoon Seol[3]

Abstract *- This paper focuses on the program development process, and how the Computer & Information Science department utilized the various program and professional requirements sources to develop a BSSE program proposal that is completely accreditable. The requirements sources referenced include the EC2000, CC2001, SEEK, SWEBOK, existing Computer Science courses, available faculty resources and the not-insignificant required university courses. Specifically, this paper describes the use of traceability matrices to map program requirements from the SEEK, SWEBOK and other sources against the courses that form the core of the BSSE program design. The paper also includes a description of various tradeoffs and program design rational used in the process.*

Index Terms – Software Engineering Education, Program Design, Program Accreditation

TASK: ACCREDITABLE BSSE PROGRAM

The task presented to the organizing committee was the development of an accreditable Bachelors of Science in Software Engineering (BSSE) program that could be successfully fielded at Gannon University. This primarily includes the organization of the program and the definition of the courses required, with the ability to use as much as possible of the current courses in use in the current BSCS, BSEE, MSCIS, MSESE programs currently in place.

Not the least daunting of the issues was simply defining what Software Engineering was as a discipline to students, the university, and ourselves. To this end, the materials available through ABET 2000 (EAC guidelines)[1], the SWEBOK [2] efforts, the Developing Software Engineering volume [3] of CC2001 [4] and other published materials on Software Engineering as a discipline [5] proved invaluable. These were important issues to confront, which ultimately became a key factor for the acceptance of a program proposal for the new BSSE program. These sources strongly support the task of defining the BSSE program proposal as accreditable, and widely recognizable as "Software Engineering."

The approach taken to solve this problem was to use the various requirements sources available, and rigorously apply traceability matrices to match program components (usually courses) to the defining program requirements. The defining requirements included multiple sources, though not all were clearly documented. Thus throughout this paper, tables are used to match the proposed program design (i.e., courses) against documented professional and educational materials that help define Software Engineering as a discipline. This paper will not cover the more lengthy discussions surrounding feasibility issues, including pedagogy, course prerequisite structures, staffing, design experiences which have also influenced the development of Gannon University's current BSSE program proposal. Note that the 'Final' draft of this proposal is by no means 'Final', but was used as the foundation for College, University and Board-of-Directors approval (still pending as of April 2003).

BACKGROUND

Gannon University is a small, Catholic University located in Northwestern Pennsylvania. Through the College of Engineering and Computer Science (ECS), the university has enjoyed offering two ABET-accredited BS engineering programs in Electrical Engineering and Mechanical Engineering since the mid-1960s, and Computer Science programs since the late 1960s. Added to these core undergraduate programs were associated Masters programs in Electrical Engineering (1970), Mechanical Engineering (1970), Embedded Software Engineering (1997) and Computer and Information Science (2001). Three departments, Electrical & Computer Engineering (ECS), Mechanical Engineering (ME) and Computer & Information Science (CIS), have supported these.

Experience with the Embedded Software Engineering [6] and Computer and Information Science graduate programs influenced the decision to pursue development of a Software Engineering undergraduate degree. Another factor was the development of an Embedded Software track within the ABET-Accredited BSEE degree [7]. Experience with these program development efforts emphasized the importance of accreditation, especially in a new and growing discipline. This in turn led to the desire to systematically apply the existing and developing curriculum guidelines, as well as other recognized sources to guide these efforts.

The basic design procedure followed here was to formulate an initial draft of the program based on known university constraints, and expected accreditation criteria, which was initiated using ABET and SWEBOK information. Then a detailed analysis was done against the draft SEEK documents, and concluded by taking a detailed look at what

[1] Steve Frezza, Dept. of Electrical & Computer Engineering, Gannon University, 109 University Square, MB3181, Erie, PA 16541 frezza@gannon.edu
[2] Sreela Sasi, Dept. of Computer & Information Science, Gannon University, 109 University Square, Erie, PA 16541 sasi@gannon.edu
[3] Jaehoon Seol, Dept. of Computer & Information Science, Gannon University, 109 University Square, Erie, PA 16541 seol@gannon.edu

0-7803-7961-6/03/$17.00 © 2003 IEEE **November 5-8, 2003, Boulder, CO**
33rd ASEE/IEEE Frontiers in Education Conference

departmental and pedagogical issues remained. Table X provides a compact view of the 'final' program proposal.

Formation of the program requirements came from five primary sources:

- College and University requirements
- ABET requirements (EAC 2001-2002)[1]
- Knowledge Areas defined by the Software Engineering Body of Knowledge (SWEBOK)[2]
- The August 2002 draft of the Software Engineering Education Knowledge areas (SEEK)[3], which is part of the Computing Curricula (CC 2001) effort [4].
- Department and program compatibility requirements

This last requirements source consists of the specific requirements and constraints needed to make the degree program easily compatible with other degree programs offered by the CIS department, specifically the core programming sequence required by all CIS majors.

Applying College and University Requirements

Developed in the Liberal-Arts College tradition, Gannon University includes a mandatory general education requirement called the *Core of Discovery*. The Core of Discovery includes 36 credits [8] in areas of English (three courses 9 Cr.), Philosophy (two courses, 6 Cr.), Theology (two courses, 6 Cr.), Social Science (one course 3 Cr.), Fine Arts (one course, 3 Cr.), History (one course, 3 Cr.), Ethics (one course, 3 Cr.) and Capstone (one course, 3 Cr.).

These courses serve to ensure that every Gannon graduate achieves the university outcomes. These courses are taken by all majors, and are not specialized courses for specific majors. ECE and CIS programs have utilized a 6-credit capstone to facilitate the integration of ethics into multi-disciplinary projects [9,10].

While the University sets the broad requirements for the inclusion of Core of Discovery courses within a program, programs are permitted to restrict these courses. In the case of our Software Engineering program proposal, the committee elected to restrict the second philosophy course to "Philosophy of Knowledge" to tailor the students' philosophical background to better support concepts foundational to requirements engineering.

Consequently, University and college requirements immediately drove the selection of 39 credits within the program, including six Computer and Information Science credits.

Applying ABET/EAC Requirements

EAC supports two broad categories of requirements that apply to the design of engineering programs. These are program-wide guidelines, and discipline-specific guidelines. Although written very simply, the EAC 2001-2002 Software Engineering guidelines touch broadly on many areas of Software Engineering. These specific curriculum requirements fit within a broader context of the requirements, commonly referred to as A-K requirements [1].

Based on the experience in the accreditation of our BSEE program, the recommending committee agreed that all of the A-K criteria would be easily supported with the initial program proposal, with the exception of Lifetime Learning. None of the existing course structures squarely addressed lifetime learning issues. Thus a pattern is adapted that is used in Gannon's BSEE program, where the program includes a specific 1-credit "Professional Seminar" course that specifically ties together the lifetime learning themes developed throughout the students' course of study.

The current analysis divided the Software Engineering-specific EAC requirements into eleven points that apply directly to the program design [1]:

The program must demonstrate that the graduates have the ability to

1. *Analyze software systems*
2. *Design software systems*
3. *Verify software systems*
4. *Validate software systems*
5. *Implement software systems*
6. *Apply software systems*
7. *Maintain software systems*
8. *Appropriately apply discrete mathematics to complex software systems*
9. *Appropriately apply probability and statistics to complex software systems*
10. *Relevant topics in computer science and supporting disciplines to complex software systems*
11. *Have the ability to work in one or more significant application domains.*

The first ten of these requirements served as general guidelines, and were easily traced to course suites in the proposed program. Table I illustrates this traceability.

TABLE I
COVERAGE OF EAC SE CURRICULUM REQUIREMENTS

EAC Topic	Proposal Coverage (Courses)
Analyze	Software Engr., Requirements, Capstone Design
Design	Software Engr., Design & Test, Soft. Architecture
Verify, Validate	Design & Test, Testing & Quality Assurance
Implement	Programming Courses, Personal Software Process
Apply	Software Engr., D&M of User Interfaces, Capstone
Maintain	D&M of User Interfaces
Discrete Math	Discrete Math 1 & 2, Formal Methods
Prob. & Stats	Probability & Statistics, T&QA, Capstone Design
Relevant CS, Other domains	Intro to Engr., Programming Courses, DB, Distributed Programming
Application Domain	Application Domain Track (focused credits – see Table II)

However, the eleventh requirement required significant consideration, and led to the development of a nine-credit 'application domain' track, wherein students take three courses outside of the Software Engineering core, and focus on an application domain. This idea fit well within the

program desires to make the degree flexible, as well as having a proven track record with other ABET-approved BSSE programs such as that at the Rochester Institute of Technology [11]. Table II indicates the suggested application domain concentrations. Each BSSE student would be required to declare and complete one of these sequences.

TABLE II
SUGGESTED APPLICATION DOMAIN CONCENTRATIONS

Domain Concentration	Course Sequence
Computer Engr.	Adv. Digital Logic, Adv. Comp Arch, VLSI Systems
Electrical Engr.	Communication, Circuits, Signals & Systems
Embedded Systems	Adv. Digital Logic, Microprocessors, Embedded Systems Lab
Computer Science	Algorithms, Comp. Languages, Operating Systems
Information Systems	Multi-media, Adv. Web Programming, Adv. Networking
Mechanical Engr.	Statics, Dynamics, Thermodynamics

In sum, the EAC requirements drove the addition of four credits specifically, with influence on most other program components.

Applying SWEBOK

While not accreditation materials, the SWEBOK proved to be a valuable resource in helping define key areas for this program development. This use was envisioned during the development of the SWEBOK document[2]. The Knowledge Areas (KA) defined proved to be easy to read and explain, easily traceable to courses, and useful in the definition of draft syllabi for new courses. Table III outlines the tracing of KAs to courses where the material would be covered.

TABLE III
COVERAGE OF SWEBOK KNOWLEDGE AREAS

SWEBOK Knowledge Area	Course Coverage
Requirements	Software Engr., Requirements, Capstone Design
Design	Design & Test, Software Architecture
Construction	Programming Courses, Personal Software Process
Testing	Design & Test, Testing & Quality Assurance
Maintenance	Design & Maintenance of User Interfaces
Management	Software Engr., Requirements, Capstone Design
Configuration Mgmt	Software Engr., D&M of User Interfaces
SE Process	Personal Software Process, Software Engr.
Tools and Methods	Software Engr., Formal Methods, Requirements, Design, T&QA Courses
Quality	Software Engr., Testing & Quality Assurance

Table III defines how the BSSE program design would cover the core knowledge areas defined by the SWEBOK. These KAs were used in the definition of the initial syllabi for new courses. As the program continues, the recommending committee felt that these KA definitions would be invaluable to instructors. They would be expected to read and be familiar with the SWEBOK KAs that affect their courses, and use them to influence proposed changes to the syllabi.

Applying the SEEK

The SEEK in its draft form proved to be the most influential (and difficult) of the core materials to be applied to the program development effort. The foundation was the August 28, 2002 draft of the Software Engineering Education Knowledge volume of the Developing Computing Curricula – Software Engineering Volume [3]. In this area, the committee had 272 specific recommendations (topics) to contend with, organized into ten knowledge areas with 40 sub areas (units). Of the 272 topics defined in the SEEK, 239 were listed as "Essential," while the remainder were listed as "Desirable" or "Optional" (E-D-O) Table IV outlines the SEEK KAs, and provides a summary of the topics, units and essential units defined.

TABLE IV
SUMMARY OF AUGUST 2002 (DRAFT) SEEK KNOWLEDGE AREAS, TOPICS AND UNITS

SEEK Knowledge Area	Topics	Units	Essential Units
Fundamentals	4	37	34
Professional Practice	3	15	15
Requirements	6	32	24
Design	7	48	44
Construction	4	31	27
Verification and Validation	5	31	28
Evolution	1	9	7
Process	2	13	12
Quality	4	26	21
Management	5	30	27
Total	**41**	**272**	**239**

Previous experience with the importance of tracing ABET criteria to courses and program goals led this committee to apply the detail provided in the SEEK knowledge areas directly to the proposed course of study. When faced with this very detailed set of requirements that the program design was expected to meet, the committee chose to use traceability matrices to highlight any requirements gaps.

The approach taken was to assign responsibility for coverage of particular knowledge units to particular courses. Each knowledge unit defined in the SEEK was assigned a primary (and occasionally secondary or tertiary) course responsible to deliver this KA. Table V illustrates a summary of the courses assigned primary responsibility for SEEK knowledge units.

Three observations can be seen from this summary:

- Not all units (in fact, not even all *essential* units) were covered by the courses.
- Knowledge units cover a broad spectrum of university courses.
- SE courses covered the majority of knowledge units

TABLE V
RESPONSIBILITY ASSIGNMENT OF SEEK UNITS ASSIGNED TO COURSES

Type	Courses	Units Covered	Cr.
CIS	Introductory Programming Sequence (four courses)	13	9
CIS	Principles of Systems	1	3
CIS	Data Structures	6	3
CIS	Intro to Networks	1	3
CIS	Adv Object-Oriented Prog	4	3
CIS	Database Management Systems	2	3
CIS	Distributed Programming	3	3
ENGR	Professional Seminar	2	3
ENGR	Intro to Engineering	1	3
ENGR	Computer Architecture	1	3
ENGR	Capstone Design	21	3
MATH	Discrete Mathematics	7	6
MATH	Probability & Statistics	2	3
SE	Software Engr Seminar	4	1
SE	Formal Methods in Software Dev.	15	3
SE	Personal Software Process	9	3
SE	Software Design & Test	19	3
SE	Software Engineering	16	3
SE	Requirements Engineering	22	3
SE	Software Architecture	23	3
SE	Testing & Quality Assurance	40	3
	Total	**246**	**73**

This responsibility-assignment (trace) approach proved to be relatively easy for existing courses, but more difficult for (typically new) courses categorized as "Software Engineering" courses. Table VI illustrates the primary and secondary assignments allocated to CS-2 equivalent course "Data Structures and Algorithm Design." A total of six primary unit coverage responsibilities and five secondary ones were assigned to this course.

TABLE VI
COVERAGE OF SEEK UNITS ASSIGNED TO "DATA STRUCTURES" COURSE

Unit	Unit Description	EDO	Trace
FND.cf	Algorithms, DS & Complexity	E	Primary
FND.cf	Control & Data Typing, Recursion	E	Secondary
DES.str	Function-Oriented Design	E	Secondary
DES.dd	Algorithm design	E	Secondary
DES.dd	Data design	E	Secondary
CON.lan	Programming Idioms	E	Primary
CON.lan	Parameterization and Generics	E	Primary
CON.tec	Selection of Data Structures and Algorithms	E	Primary
CON.tec	Code Reuse and Libraries	E	Secondary
CON.tl	Development environments	E	Primary
VNV.par	Debugging/fault isolation techniques	E	Primary

Assigning SEEK unit responsibility to new courses proved to be more difficult. The primary issue was that of *Over Specification*. Many knowledge topics were found to be described in more detail than what is used in defining course outlines. Table VII outlines the primary coverage for the "Software Testing and Quality Assurance" course, which is one of the new Software Engineering courses defined for the program. The overabundance of detail can be seen in the 42 listed knowledge units, and the committee are not hopeful that these will be reasonably covered in a three-credit-hour semester long course. The detailed course outlines (developed from these traceability lists) were almost uniformly pared down to manageable levels.

TABLE VII
COVERAGE OF SEEK UNITS ASSIGNED TO "TESTING & QUALITY ASSURANCE" COURSE

Unit	Unit Description	EDO
REQ.va	Acceptance test design	E
REQ.va	Verifying quality attributes	E
DES.ev	Evaluation Criteria	E
DES.ev	Evaluation Techniques	E
DES.ev	Design measurement and metrics	E
CON.tl	Unit-testing tools	E
VNV.fnd	Documenting V&V strategy	E
VNV.fnd	Reliability Metrics	E
VNV.fnd	V&V Involvement at different points in the lifecycle	D
VNV.tst	Developing test cases based on use-cases	E
VNV.tst	Operational profile-based testing	E
VNV.tst	System and acceptance testing	E
VNV.tst	Testing across quality attributes	E
VNV.tst	Regression Testing	E
VNV.tst	Testing Tools	E
VNV.tst	Deployment process	D
VNV.par	Analyzing failure reports	E
VNV.par	Defect Analysis	E
VNV.par	Fault Tracking	E
PRO.con	Modeling and specification of software processes	E
QUA.cc	Society's concern for quality	E
QUA.cc	The dimensions of quality engineering	E
QUA.cc	Roles of people, processes, methods, tools and technology	E
QUA.cc	Quality models and techniques	D
QUA.std	The ISO 9000 series	E
QUA.std	ISO/IEEE standard 12207: the "umbrella" series	E
QUA.std	Tailoring and adaptation of standards	E
QUA.pro	Software quality models and metrics	E
QUA.pro	Root-cause analysis and defect prevention	E
QUA.pro	Quality-related aspects of software process models	E
QUA.pro	Introduction/overview of ISO 15504 and the SEI CMMs	E
QUA.pro	Quality-related process areas of ISO 15504	E
QUA.pro	Quality-related process areas of SW-CMM and CMMIs	E
QUA.as	The nature of process and product assurance	E
QUA.as	Distinctions between assurance and V&V	E
QUA.as	Quality planning	E
QUA.as	Organizing/reporting for process/product assurance	E
MGT.ctl	Techniques of process and product assurance	E
MGT.ctl	Monitoring and reporting	E

The application of the SEEK analysis affected both the Computer Science and Software Engineering components of the program proposal. However, the most profound impact of the SEEK was in the selection and intended contents of the Software Engineering courses. However, while recommended, the detail provided by the SEEK knowledge units are not explicitly required for accreditation [12], so the recommending committee will revisit this analysis in the future. Table VIII illustrates the changes to the Software Engineering-specific courses that occurred from this analysis.

Essentially, there were three changes: The stipulation was that the CS and SE students take similar courses for their first two years of study; For pedagogical reasons it was found useful to insert a one-credit seminar to help orient

freshman SE majors to Software Engineering, and help support them in developing as SE professionals.

TABLE VIII
CHANGES TO SE-SPECIFIC COURSES DUE TO SEEK ANALYSIS.

Early Draft (24 Cr.)	Final Draft (25 Cr.)	Rational
	Software Engr Seminar	Pedagogy
Requirements & Formal Specification	Formal Methods in Software Development	SEEK Emphasis
Personal Software Process	Personal Software Process	-
Software Design & Test	Software Design & Test	(current course)
Software Engineering	Software Engineering	(current course)
Economics of Software Development	Requirements Engineering	SEEK Emphasis
Software Architecture	Software Architecture	-
Software Testing & Quality Assurance	Software Testing & Quality Assurance	-
Software Maintenance & Evolution	Human Interface Design & Maintenance	SEEK Emphasis

The second change was the strong emphasis on Formal Methods found in the SEEK. The original approach was to introduce Formal Methods as a two-three week unit within the context of a requirements/specification course. However, the amount and depth of the Formal Methods material listed in the SEEK influenced the inclusion of a full course coverage of formal methods. Despite this addition, we do not believe that even a three credit semester-long course will permit students to have adequate coverage to the extent that the SEEK contributors communicated. This in turn led to a reconsideration of the requirements material intended for the original "Requirements and Formal Specification" course, which was expanded to a three-credit course on requirements engineering. Credit restrictions forced the committee to abandon its original intent of including a course on Software Engineering economics.

The third major change to the SE core program was the restructuring of the original "Software Maintenance and Evolution" course. The committee revised this course definition to use User Interface design & development as its domain. While this makes the course very specialized, the potential instructors (and the recommending committee) felt that the course would still be feasible, practical and extremely interesting. A more careful consideration was in the proper inclusion of course prerequisites so that students would have sufficient background in UI technologies before taking the course, and to make it available as a technical elective to students in other programs. This dovetailed well with foundational programming constraints worked out to ensure compatibility with other department programs.

Applying Department & Program Compatibility Requirements

The last influential set of requirements that affected the design of the program was the requirement to produce a program that was compatible with the existing computing programs currently offered by the CIS department. The first observation (strongly influenced by the experience with CC 2001) was that the introductory/core CS material was very similar between the CS and SE programs. The faculty were unanimous in seeing overlapping need for solid, practical foundations in programming, systems, networking and distributed systems, as well as introductory design and testing, and software engineering topics. Furthermore, these programs are offered to relatively few students. Hence there is desire to make the programs as alike as possible, both from a scheduling/staff resource perspective and from the desire to permit students to switch majors early in their academic career without forcing them to extend their tenure at the University.

These issues weighed heavily against the desire to have early project experiences for SE students. The compromise reached was to have an early "SE Seminar" course to better prepare SE students for the context in which they learn computing fundamentals, and support them to integrate SE foundations into the computing foundations courses shared by all students (CS/MIS/SE) in the department. This included a careful restructuring of the material in the introductory programming sequence. Table IX outlines the changes to the Computer Science component of the program.

TABLE IX
CHANGES TO CS-SPECIFIC COURSES TO SUPPORT PROGRAM COMPATIBILITY.

Early Draft (22 Cr.)	Final Draft (33 Cr.)	Rational
	Database Management	Course Pre-req
Intro to Computing	Intro to Computing (CS0)	(current course)
	Visual Programming	Course Pre-req
	Principles of Systems	Program Req
Introduction to C++	Introduction to C++ (CS1)	Restructured
	Problem solving in C++ (CS1&2)	Pedagogy/PR
Data Structures	Data Structures (CS2)	Restructured
Intro to Networks	Intro to Networks	(current course)
Advanced OOP	Advanced OOP	(current course)
Database Mgmt Systems	Database Mgmt Systems	(current course)
Network Programming	Distributed Programming	(current course)

This traceability exercise led to positive benefits beyond the BSSE program development process – it helped fuel significant and positive changes in related curricula. These course sequence changes were considered so vital to the department programs, that the course restructuring and associated catalog revisions to the Computer Science and Management and Information Systems programs proceeded independently of the acceptance of the BSSE proposal.

ANALYSIS

What was observed from these traceability exercises was that detail of the SEEK units was somewhat overwhelming, which left a program (particularly a more constrained program such as the proposed one) little room to focus on issues in Software Engineering that the committee wanted the students to have more experience with. A case-in-point was the initial desire to include a course-long focus on Software Engineering Economics and its role in business issues, which was subsequently replaced by a course on Formal Methods (see Table VIII and discussion). These

issues will be revisited if the program draft is approved. Table X outlines the final proposal in its final form, matching the course(s) against the number of semester credit hours allocated.

TABLE X
GANNON BSSE PROPOSAL AT-A-GLANCE

Engineering		Mathematics & Science	
Introduction to Engineering	3	Calculus	6
Computer Architecture	3	Discrete Mathematics	6
Professional Seminar	1	Probability & Statistics	3
Capstone Design 1 & 2	6	Physical Sciences & Labs	8
Computer Science		**Software Engineering**	
Database Management	1	Software Engr Seminar	1
Intro to Computing (CS0)	3	Formal Methods	3
Intro to Visual Programming	3	Personal Software Process	3
Principles of Systems	3	Software Design & Test	3
Introduction to C++ (CS1)	3	Software Engineering	3
Problem solving in C++ CS2)	3	Requirements Engineering	3
Data Structures	3	Software Architecture	3
Intro to Networks	3	SW Testing & Quality Assurance	3
Advanced OOP	3	Human Interface Design & Maint.	3
Database Mgmt Systems	3	Application Dom. Concentration	9
Distributed Programming	3		
Core of Discovery			
Writing	6	Literature	3
Social Science	3	Theology	6
Intro to Philosophy	3	Philosophy of Knowledge	3
History	3	Fine Arts	3
Phil/Theo Moral Responsibility	3		

While other sources (such as SWEBOK and ABET) left significant maneuvering room for program development, the SEEK strongly influenced the committee to remove "Important" SE program material, and replace it with SEEK "Essential" program material. Similarly, the traceability exercise led to a very detailed look at outcomes expected department-wide and service courses. Most of this scrutiny was worthwhile, and led to changes in the CS and MIS programs, and a commitment from the Math department to offer a second discrete math course. The side effect was that inclusion of useful CIS courses removed the possibility of open or technical electives for BSSE majors.

While the observation that the SWEBOK and ABET criteria are useful, yet flexible, the experience in rigorously applying the SEEK was not the same. The SEEK draft was very useful as a defining document. Table XI illustrates the KA-level comparison of the SEEK and SWEBOK.

SEEK clearly defines what are the important SE knowledge areas, and provides reasonably clear supporting details as to what different topics are to be considered. For example, SEEK includes areas in "Professional Practice," which are not part of the SWEBOK or as clearly handled in the current ABET criteria.

However, the detail provided at the unit level of the SEEK seemed to be a two-edged sword for program design. With the rigorous application of the unit-level information, the recommending committee agreed that the credit hours needed to cover the 'essential' units would be well beyond the department's ability to offer a program within University constraints. This overwhelming amount of essential detail was too detailed to be used for course planning, and often obscured what units are more essential than others.

TABLE XI
COMPARISON OF SWEBOK AND SEEK KAS

SWEBOK Knowledge Area	SEEK Knowledge Area
Software Requirements	Software Requirements
Software Design	Software Design
Software Construction	Software Construction
Software Testing	Software Verification & Validation
Software Maintenance	Evolution
Software Architecture	Software Architecture
Software Engineering Management	Management
Software Configuration Management	Subtopic of Management KA
Software Engineering Process	Software Process
Software Engineering Tools and Methods	*No Equivalent – embodied in KA topic details*
Software Quality	Software Quality
No Equivalent	Professional Practice

REFERENCES

[1] *Engineering Criteria 2000 3rd Edition: Criteria for Accrediting Programs in Engineering in the United States*. Published by the Accreditation Board for Engineering and Technology (ABET), Baltimore MD. 1997. Available at http://www.abet.org/EAC/eac2000.html.

[2] Abran, A. and Moore, J, Eds., "Guide to the Software Engineering Body of Knowledge", Version 1.0, IEEE Press, May 2001. See www.swebok.org.

[3] Sobel, A. (Ed.), "Computing Curricula – Software Engineering Volume Software Engineering Education Knowledge (SEEK) 1st Draft," August 28 2002. Available at http://sites.computer.org/ccse

[4] *Computing Curricula 2001, Final Draft*, December 2001. Available at http://www.computer.org/education/cc2001

[5] Diaz-Herrera, J., "Engineering Design for Software: On Defining the Software Engineering Profession," *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference*, November 2001 Reno, NV.

[6] Frezza, S., "Integrating an Industrial Practicum into a Graduate Embedded Software Engineering Program," *Proceedings of the 29th ASEE/IEEE Frontiers in Engineering Education Conference (FIE '99),* San Juan, PR, November 1999.

[7] Frezza, S., "An Undergraduate Software Engineering Program … in Electrical Engineering," *Proceedings of the 28th ASEE/IEEE Frontiers in Education Conference*, November 1998, Phoenix, AZ.

[8] *Gannon University Undergraduate Catalog 2002-2003*, Gannon University Press, Erie, PA

[9] Voshall, R. and Frezza, S., "Capstone for Cross-Disciplinary Ethics and Design," *Proceedings of the ASEE 1999 Spring Conference, North-Central Section*, Erie, PA April 1999.

[10] Frezza, S., and Voshall, R., "Industry/Academic Collaboration in Senior Design," *Proceedings of the ASEE 1999 Spring Conference, North-Central Section*, Erie, PA April 1999.

[11] See http://www.se.rit.edu/ or http://www.se.rit.edu/appdomains.php

[12] Lidtke, D., Leone, J., and Gorgone, J. "Update on the Accreditation of Computing Programs", Panel Session in *Proceedings of the 32nd ASEE/IEEE Frontiers in Engineering Education Conference (FIE '02), Boston, MA*