Review Comments for
CCSE Volume (first draft- 7/17/03)

This document represents the response to reviewer comments to a public call for review of the CCSE Volume (draft 7/1/7/03). The review period extended from July 2003 through September 2003. Each comment is paired with a response form the CCSE Steering Committee. In most cases the reviewer comment is quoted verbatim; however, in some cases the comment was summarized or abbreviated to improve document readability. There were a number of comments about grammar and style that were not included below, but were used to improve the CCSE Volume. The CCSE Steering Committee is appreciative of the careful and thoughtful review by the reviewers; we feel their comments and suggestions have produced a much-improved document.

reviewer	200017959 - William Griswold
comment	I have deep concerns about drawing a Science/Engineering distinction between Computer Science and Software Engineering.
CCSE response	There has been great debate about this issue; but the decision to have separate CS and SE volumes partly motivates the position taken in the SE volume. Some wording was changed in Chapter 3 to better emphasize the relationship between CS and SE.

reviewer	2000184579 - Helen M Edwards
comment	Very clearly written, giving a coherent rationale for the need for the volume and the mindset needed by those who wish to develop u/g SE programmes. What I particularly like is that the volume gives an impression of self-confidence in SE as being truly an engineering field. Too often in the past SE has been defined in terms of computer science and aspirations to become an engineering discipline. I think that the current document is as clear as it can be about the variations that may exist in other countries. I am not sure further adaptations would be beneficial (the eventual documentation would be so complex that the benefit of additional adaptations may well be lost).
CCSE response	Noted. We feel the adaptations are helpful and extra ones are provided, since they are only very simple grids in Chapter 6.

reviewer	1928432 – Matthias Felleisen
reviewer comment	Need to add the semantics of programming languages
CCSE response	Modifications made. We agree with the importance of understanding the semantics of

	programming languages. It is unfortunate that this topic is not even part of the required core of the CCCS. It is now listed as one of the desired topics in the CMP KA.

reviewer	2000209673 - Larry Bernstein
comment	The emphasis on quantitative analysis is good. The focus on ethics, teamwork and human communications is on target. More emphasis on sampled data control systems is needed including feedback control analysis using Z-transforms or Discrete time math or eigenvalues. For Requirements- Quality Function Deployment For Architecture- Simplification Techniques and Metrics and the reliability equation. For Design- Sampled Data Feedback Control For Testing- Orthogonal Array Theory and its application to test case reduction For Maintenance- Structure analysis to simplify build or execution design using visualization techniques
CCSE response	Control issues are part of the “Specialties and Their Related Topics” of the SEEK. We did not feel it should be essential knowledge for all programs, but some specializations would require it.

reviewer	20002127960 - Robert L. Glass
comment	4.8 I don't see recursion as a programming fundamental. OO run-time issues seem a peculiar topic, given that methodologies in general have not been mentioned here. "Hot spot" analysis is too limiting. This should be "performance analysis tuning."
CCSE response	Recursion appears in CMP.cf.1. The emphasis on OO run-time issues stem from the interpretation of polymorphism. The last issue has been incorporated.
comment	4.8 Automatic generation of code is hardly worthy of inclusion. It's been identified for more than a decade as a "cocktail party myth."
CCSE response	Such tools do indeed exist; however, their application is indeed very limited.
comment	4.9 Studies of practitioner math needs show this area to be needing further research analysis. I would suggest that this whole section be labeled "work in progress" until that research has been completed. Empirical/experimental approaches should be about more than "CPU and memory usage."
CCSE	Given what we know today, this section is appropriate. If future

response	analysis shows that this KA is not appropriate, then it will surely be rewritten. As with almost all of our parenthesized phrases, the items listed are meant to represent examples, not an exhaustive list.
comment	4.12 The "design for..." quality attributes should include maintainability.
CCSE response	Maintainability is certainly an example of a quality attribute.
comment	4.18 The financial/e-commerce section has no sub-topics on e-commerce. The safety-critical section implies that formal/proofs are the only way to ensure safety. (a) this has not been shown to be true, and (b) there are lots of other vital techniques. Embedded/real-time places too much emphasis on hardware. Under scientific systems, it should list "depth in applied mathematical approaches."
CCSE response	The specialty areas are certainly not meant to be detailed (even the list of potential areas is not meant to be complete). We strived to list only a couple of pertinent topics for each area.
comment	4.16 The quality attributes should explicitly include maintainability.
CCSE response	The description has been changed to include maintainability – it was added.
comment	5.2 Somewhere in this section, there should be a discussion of teaching "programming in the large" when classroom exercises can only be "programming in the small." This could be under "outcomes," where an outcome should be the ability to program in the large event when the student has only marginally tried out this skill.
CCSE response	Added as a recurring theme in guideline 4.
comment	6.2 Somewhere here there should be a discussion of the merits of teaching "reading before writing." We do that in all other language fields, yet we resist it in computing/software.
CCSE response	An adjustment to guideline 4 has been added in response to this. Also a recommendation has been added to SE200 and SE201 that students work with existing systems.

reviewer	20002216441 - Garth Glynn
comment	I think that some of the introductory sections, viz. chapters 3 and 5 contain some excellent material. I am particularly in favour of the guidelines in 5.1
CCSE response	Noted.

reviewer	20002316684 - Luisa Mich
comment	2.1 Items 5 and 6: couldn't these be merged
CCSE response	We agree that these principles are related, but feel address distinct concerns: [5] speaks to curriculum structure and [6] to the knowledge on which the curriculum would be based.
comment	2.1 Item 9: must be multi-perspective instead of must be broadly based
CCSE response	By the term "broadly based" we mean not only multi-perspective, but based on various stakeholders that may have a similar perspective (e.g., come from different sectors, as indicated in the second sentence).
comment	2.2 item 2: quality executable artifacts instead of executable artifacts
CCSE response	Item 2 was changed to reflect "quality artifiacts".
comment	2.2 Name the items, e.g.: [1] Core knowledge, [2] Individual-group work [3] Project management [4] Multi-concerns design [5] Change management/empirical-defined process models (split this point in two) [6] Relational abilities [7] Long-life learning
CCSE response	We are not sure of the value of using such labels; references in other parts of the document using section and outcome number seems to work fine. However, we have added additional text to each outcome to elaborate on the meaning of the outcome.
comment	3.2.1 Figure 1 - box the two parts before and after the arrow Couldn't it be useful to add something more about e-learning experiences (or platforms) benchmark for SWE
CCSE response	The figure has been eliminated.

reviewer	20002629178 - Vladan Jovanovic
comment	CMP.f11 Add SQL and XML change c to a
CCSE response	This topic is listed in the same level of detail and at the same Bloom's level as networking.

reviewer	2000278704 - Dino Mandrioli
comment	Positive Comments All in all, I like it: I like the rationale, the principles, the guidelines, the structure, and the presentation.

	Negative Comments I am rather unhappy with the way you address the issue of SE within the whole engineering context. As a premise, let me say that I very much agree (at least 90%) with Ilia Bider comments to previous drafts, which could be summarized as “there is too much SW and too little Eng.” Thus, I do not repeat his (her) arguments.
CCSE Response	There has been great debate about the issue related to the proper balance between CS, SE and general engineering knowledge and pedagogy. We have tried to strike an appropriate and useful balance for undergraduate SE programs.
comment	… if one has learned the essence of analysis, there is no need to devote so much time to requirements elicitation, specification, validation, then again to V&V, etc. I agree that a serious, though difficult, application of curriculum guidelines 4 and 18 can strongly mitigate this criticism but I also ask whether courses such as SE321, SE322, SE323, etc. are good examples of application of these guidelines.
CCSE Response	We believe that all courses should be able to incorporate the kind of interdisciplinary approach and synergy expressed in these guidelines.
comment	Finally, a few words about Professional Practice. Here I have mixed feeling. I agree that there are important “non technical” skills, such as communications skills, that must be possessed by, and -one way or another- taught to, everybody (can you imagine a lawyer who cannot participate actively in a meeting); but I wonder how much of such skills can and should be taught and learned in a university curriculum. … I also acknowledge that such skills are often lacking in most students and even in various professionals and therefore something has to be done to fill up this hole; but, to give a concrete example, I seriously doubt that devoting 20 class hours (!) to issues such as accreditation, code ethics, professional societies, etc. will do a better job than simply emphasizing the relevance of such issues in 2 hours. Thinks less time could be spent on many professionalism topics.
CCSE Response	There is disagreement on this item. Hopefully, the new course descriptions in Appendix A will show the value of this material.

reviewer	20002827999 - Deepak Dahiya
comment	A 1 unit credit optional course on an management module needs to be included The course can have topics ranging from organization role, cross -culture work environments, outsourcing that all Software Engineers cannot ignore in this global industry.
CCSE Response	The CCSE focuses on the core material needed in an undergraduate SE program; hence, there are many potential

	optional topics that have been left out. However, at their option, designers of SE programs could easily incorporate the reviewer's recommended material in some of the course that are outlined (e.g., NT272 Engineering Economics or NT291 Professional Software Engineering Practice).

reviewer	2000292630 - Jonathan Hodgson
comment	First I think that some material on security, particularly design for and coding for security needs to be in the core. Places where you might put it are QUA, FND, DES or PRF. Or perhaps all of these. I know it appears in FND 4 but very much in passing. It is more important than that.
CCSE Response	While security issues have been increasing in importance, little is found in current undergraduate curricula on it (which most likely explains why little to no reference has been found in other KAs). As the exposure to security issues increases in the classrooms, it will also be found in more KAs.

reviewer	2000306639 - Duncan Hall
comment	3.3.1 Last sentence to include wording such as "desire to seek verifiable quantitative data on which to base decisions yet also able to function effectively in an environment of ambiguity"
CCSE response	Good point – incorporated in section 3.3.1.
comment	3.3.2 Some coverage of the dilemmas facing professional engineers in employment situations such as ethics and implications of "whistle blowing" might be useful for example.
CCSE response	Agree – incorporated in last paragraph in section 3.3.2
comment	3.3.3 Need to address the need to be aware of the limitations of "formula-based" modeling.
CCSE response	Good point – incorporated in section 3.3.3.

reviewer	20003129876 - Maurizio Fenati
comment	… add 'Optimization Models & Theory basics' reference as well as 'Mathematical Analysis basics' one within 'Mathematical and Engineering Fundamentals' area.
CCSE Response	The topic 'mathematical analysis' can be found in the listed mathematics courses. An operational research topic as 'optimization models & theory' was not judged to be necessary knowledge for all software engineering graduates.
comment	It should be considered 'Business Modeling' discipline, even if I'm

	not sure if 'Software Modeling and Analysis' area will fit. According to RUP (standard 'de facto') this discipline describes how to develop a vision of the new target organization, and based on this vision define the processes, roles, and responsibilities of that organization in a business use-case model and a business object model.
CCSE Response	It is not clear what is being asked to be added where but 'Business Modeling' is mentioned under MAA.tm.6 – Enterprise modeling.

reviewer	2000322716 - Tony Cowling
comment	3.2.1 item 3 - Either here, or as a separate item in this list, reference should be made to the importance for engineers of recognizing and using good approximations.
CCSE response	agree – recommendation incorporated
comment	3.2.1 item 12 - Add a forward reference to section 3.3
CCSE response	There has been extensive revision to this section and item 12 is not longer in the list.
comment	3.2.2 Add at least some discussion of whether implementation should play such a large role in SE as is indicated here, or whether it is likely to continue to do so.
CCSE response	There has been extensive revision to this section and there is now not an emphasis on "implementation".
comment	3.3 Add near the start of the section some explanation of the significance of the engineer being a professional - viz, that it requires them to apply specialist knowledge on behalf of members of society who do not themselves have that knowledge.
CCSE response	Agree – recommendation incorporated
comment	6.3 Another guideline is needed here, concerned with emphasising to students the importance of recognising the limitations of current knowledge of SE, and of applications of that knowledge to the real world (meaning here both limitations in general, and their individual limitations).
CCSE Response	Agree - this issue has been addressed with changes to guideline 8.
comment	6.3 Another guideline is needed here, concerned with emphasising to students the importance of evaluating, challenging and seeking to improve on statements of "received wisdom".
CCSE Response	Agree - this issue has been addressed with changes to guideline 8.
comment	7.3.3 Add more discussion of the relative merits of group or individual projects. Add more discussion of the relative merits of internal or external clients.

CCSE Response	These issues have partially been addressed in the Appendix A description of the capstone course. A section on “Additional teaching considerations” has been added.
comment	7.4.2 Add a reference to the importance of calculus in any form of optimisation.
CCSE Response	Agree - change made
comment	Appendix A - Here (or earlier) it would be helpful to also have a mapping for each package defined in 7.2 or 7.3, to show how these packages cover the SEEK (and hence what they do not cover).
CCSE Response	Each of the courses in Appendix A includes information about SEEK coverage and the curriculum patterns are intended to show examples of complete SEEK core coverage.
comment	Appendix A - Here (or earlier) for each course the "learning objectives" that are referred to in the note must give as much emphasis to what bits of SE students are able to do when they have completed it, as to what they will know.
CCSE Response	Agree - learning objectives added to course descriptions in Appendix A.

reviewer	20003328180 - Andrey A. Terekhov
comment	This is a great basis for development of software engineering education programs. In particular, SEEK is a wonderful reference for designers of courses and curricula.
CCSE response	Noted.
comment	I believe that the only deficiency of the volume at the time of writing is the lack of evidence about the true value of various approaches to teaching software engineering.
CCSE response	This is good point. Unfortunately, the state of software engineering education research is relatively immature. We do site a number of papers in the Bibliography about SE education; but there is very little rigorous, definitive, widely accepted material. The information here is based on expert judgment ... as people implement the suggestions, undoubtedly flaws will be found, to be corrected in future iterations.
comment	One of the major issues in adapting Computing Curricula documents to Russian realities is the difference in duration of the education programs - typical Russian student in technical sciences graduates from the University after five years of study with a degree that is closer to M.Sc. than to B.Sc. and is called "specialist with a degree".
CCSE	We would be glad to incorporate a suggested plan using the

Response	courses provided here.
comment	Section 5, guideline 6 - Specifically mention system programming and other CS-related topics as possible domains for specialization of software engineers. This change is related to the global question of relationship between Computer Science and Software Engineering, which (in my opinion) is not properly highlighted in this volume.
CCSE Response	The CCSE volume has been revised to emphasize on the relation between CS and SE
comment	Section 6.2 Add discussion of various approaches to teaching mathematics in the introductory sequences
CCSE Response	We feel that the discussion of mathematics in section 6.2.3 is sufficient. We do not feel that a detailed discussion mathematics pedagogy is appropriate at this point in SE curriculum development.

reviewer	20003418575 - Carl J. Mueller
comment	The standard and proposed curriculums generally put a minimal emphasis on “continuous mathematics” and other “hard sciences”. I feel this is a major mistake.
CCSE Response	The CCSE is meant to guide and support a variety of programs, with different foci and domains of emphasis. Some patterns support a stronger background in mathematics and hard science than others. We suggest calculus is useful for many domains and should generally be taught in most adaptations.
comment	It is understandable that there would be differences in the four computing disciplines (Computer Science, Software Engineering, Computer Engineering and Information Technology), but there are many places where these programs intersect. Using a common nomenclature would permit instructors and authors to more easily identify the common components of these programs.
CCSE Response	We have used material and notation from the CCCS volume, but there is currently no detailed coordination for all the volumes.
comment	The thing that most concerned me about this proposed curriculum standard is the organization of Software Engineering body of knowledge within the draft.
CCSE Response	Hundreds of SE practitioners and educators were involved in the development and review of the SEEK. Major revision at this point would be difficult.
comment	Also, I could not find any direct mention of metrics in the standard. It is important that students have a basic understanding of the different types of metrics that can be derived from very simple measures.
CCSE	Metrics and measurement are listed throughout the SEEK and are

Response	incorporated in several courses.

reviewer	2000359873 - Tim H. Lin
comment	5.3 Curriculum Guideline 15: Ethical Concerns, and the notion of what it means to be a professional, should be raised frequently. => Ethical, legal, and financial (economic) concerns, and the notion of what it means to be a professional, should be raised frequently.
CCSE Response	Agree – change made.
comment	5.3 Add a new Guideline on design and coding style, in particular naming convention, coding standard. Add a new Guideline about teaching software portability. Add a new guideline about how capstone projects are conducted and also how to evaluate the success of capstone projects.
CCSE Response	We feel that these topics are too specific for curriculum guidelines. Such issues are addressed in the SEEK and the curriculum/course content sections.

reviewer	20003627382 -Tony Wasserman
comment	3.4 Add additional historical background to give a more complete and more accurate picture of the origins of software engineering education. Cite early workshops, including D.A. Oakes (et al.), SE Education: Proc. IBM Scientific Symposium, IBM Canada, 1975, and A.I. Wasserman and P. Freeman (eds.) Software Engineering Education: Needs and Objectives, Springer Verlag, 1976. Cite first paper laying out foundation for SE Education: Freeman, P. , A. I. Wasserman , R.E. Fairley, Essential elements of software engineering education, Proc. 5th Int. Conf. on Software Engineering, San Francisco, 1976, pp.116-122. Cite first paper laying out proposed curriculum for SE Education: Freeman, P. and A.I. Wasserman, Proposed Curriculum for Software Engineering Education, Proc. 3rd Int. Conf. on Software Engineering, Atlanta, 1978, pp.56-62. Cite IEEE Computer Society SE curriculum development effort circa 1978; information available from Prof. David Rine at George Mason University (drine@cs.gmu.edu)
CCSE response	Agree – recommendation incorporated.

reviewer	3721379 – Robert L. Ashenhurst
comment	Blurred distinctions between requirements, specification, and design in the Software Modeling and Analysis KA. Reviewer

	proposed that each use of the term “requirement”, “feature” and “model” in this KA be prefaced with either the word “informational” or “operational”.
CCSE response	Noted. After the third review of the SEEK, we determined that all types of modeling would be moved to the Software Modeling and Analysis KA, independent of when and under what circumstance the modeling was being performed. While other partitioning of the types of models can be made, the Steering Committee, with input from an IFIP working group, felt that this grouping was the best approach.

reviewer	20003814659 - Rob Hasker
comment	2.2 (p. 10) Soften outcome [4]; keep the bit about graduates being able to "design appropriate solutions ... using engineering approaches" but revise the bit about integrating ethical, social, and other concerns.
CCSE response	Although the “integrating” part is difficult we feel that it is important and is consistent with various accreditation criteria.
comment	3.2.1 (p. 13) Change "may be significantly wider than in other branches" to "are at least as broad as other branches".
CCSE response	There has been a major rewrite of this section and the above-mentioned phrase is no longer in the document.
comment	3.2.1, Fig. 1 Add more notation to show the proposed relationships.
CCSE response	The Figure has been eliminated.
comment	5.1 (p. 37) Revise the recommendation for certification in the third bullet item under "curriculum designers and instructors should therefore".
CCSE response	CSDP was explicitly introduced to address that problem. The Guideline 1 elaboration was modified to say "... have, or work towards", since otherwise some smaller programs would have a hard time getting established.
comment	5.1 (p. 37) Soften the prohibition against introducing bias for particular applications of SE in early courses. This is in the first bullet item under "Failure to adhere...".
CCSE response	Agree – recommendation incorporated.
comment	5.3 (p. 40) Curriculum guideline 10, first paragraph: clarify what the alternative is.
CCSE response	Agree – recommendation incorporated.

reviewer	2000394949 - Vincent Chiew
comment	Maybe this document can provide some guidance to hiring and acquiring quality instructors to achieve the aim the goal of CCSE Maybe this document can provide equivalency of qualified course so the appropriate instructors can be hired Note: The knowledge quality learnt depends largely on the instructor teaching the courses.
CCSE response	We agree this is a problem; however, most of such material is out of the scope of this document – some of this is addressed in chapters 6 and 8.
comment	Maybe this document should mention possible customization and tailoring of its contents as apply to the different businesses, industries, countries, and management styles. A SE graduate should be fully equipped with sufficient knowledge to adapt and accommodate any work environment presented or encountered.
CCSE response	We believe this is adequately expressed (within the scope of the document). Additional programs for countries have been added in chapter 6.
comment	3.2.1 [1]& 5.3 Guideline 10. Engineering is about satisfying customer needs.
CCSE response	Agree – recommendation incorporated Note: Could replace “solving customer problems” by “solving customer problems and satisfying stakeholder needs”
comment	3.2.1 [13] Add to that, the business and industry knowledge
CCSE response	“methods, techniques and technology” is mean to be domain independent and certainly could and should include “business” elements. Note: Maybe the intent of the comment is to say engineers should have a “business/industry” orientation – this should be a separate characteristic.
comment	5.1 Failure to adhere to this principle ... Bullet point # 3: It should be noted that SE should not claim to be knowledge expert in other knowledge domain too.
CCSE response	Agree – recommendation incorporated.
comment	Chapter 5 - Please note that these days many new SE graduate student might not have a SE undergraduate degree. The curriculum should either consider this situation or make it a pre-requisite such as having the student from another discipline to take some basic SE core courses
CCSE response	The CCSE Volume is focused on undergraduate SE programs.
comment	General - Maybe an institute can consider having their SE curriculum approved by IEEE by having IEEE/ACM CCSE seal of approval.

CCSE response	Accreditation issues are addressed in chapter 8.
comment	7.1.2 Please consider mature student returning to school not only for upgrade but as part of their professional development.
CCSE response	Professional development is not part of the scope of the CCSE. The focus is on undergraduate degrees.

reviewer	2000416971 - Cem Kaner
comment	The assertion that we will give students real-world experience in a classroom is implausible. We give them artificial experiences that may be educational but that do not begin to approach the complexity, time commitment, or consequences for weak work as occur in the real world. Based on the faulty assumption that we will give students "real world" experience, the curriculum assumes that the students will reach a level of maturity to appreciate heavyweight process models. Most will not. They might memorize them. Some might adopt them as an act of faith. But others will learn contempt for the heavyweight processes because they are being taught out of context.
CCSE response	We agree to an extent. However, the document (particularly guideline 14) attempts to increase the level of real-world exposure, which is better than having none. Also, there is no attempt to focus on “heavyweight processes” (or for that matter on agile processes).
comment	2.2 Delete student outcome, "Negotiate, work effectively, provide leadership where necessary, and communicate well with stakeholders in a typical software development environment.”
CCSE response	The outcome is changed so that is appropriate for an undergraduate.
comment	The document stresses too much that software engineering is a defined profession. It is not. We have widespread disagreement on the body of knowledge, the choice of lifecycle, and several other process decisions. Within the drafting group, there was sharp disagreement over whether software testing should even be taught, for example. The IEEE process standards have limited impact in most commercial software development, and less recognition, but are presented here as core process standards. I think it is widely believed among practitioners that software engineering is a craft that is perhaps on its way to gradually becoming a profession. A curriculum guide should not push an essentially political position that adopts either of these positions (or another).
CCSE response	There has been much debate over this issue and there have been some changes in Chapter 3 that partially address these concerns.
comment	4.9 – Add calculus to required core material.

CCSE response	While it is true that most NA programs require calculus, there are many other programs that do not. In particular, we do not find evidence from the topics listed in the SEEK that one year of calculus study is required. That said, we do list calculus as one of the possible university required mathematics courses.
comment	4.10 – Rework the section titled “professionalism”
CCSE response	One modification made. We believe the topics of professional societies, certification, and licensing should be known by students. The suggestion of adding “employment contracts” is a valuable addition to this list.
comment	4.13 – Reconsider how measurement is taught
CCSE response	Modification made. The topic of “criteria of valid measurements” was added as FND.ef.6
comment	4.13 – Drop the section VAV.fnd; the time would be better spent teaching people the methods of actually doing it.
CCSE response	“Teaching people the methods of actually doing it” still requires some level/amount of introduction to the method. This unit is not meant to be treated as a single model/lecture but as an introduction to the salient topics in this particular knowledge area. When the introduction is provided will depend upon the structure of the actual module/course taught.
comment	4.14 – Eliminate the section on software evolution
CCSE response	Noted. While only 10 hours of core hours of material has been devoted to evolution, this KA is essential knowledge that an undergraduate student must possess. It is true that the amount of time limits the depth and complexity that a student might experience with this KA, we believe that applications of this material might be incorporated within other projects. The selection of material and number of hours was provided by a renown expert in software maintenance, so the Steering Committee will support his judgment.
comment	4.15,4.16 Cut back on the process training
CCSE response	The amount of material devoted to the KAs of Software Process and Software Quality certainly reflects the inability for student’s to gain a rich set of experiences using such processes at the undergraduate level. Since this is most likely a student’s “first” exposure to disciplined software development, it is important that they are aware of and potentially use/apply some portions of process models and the creation of work products with certain quality attributes.
comment	4.17 – Reduce software management to a weak section on status reporting and a strong section on configuration management.
CCSE response	The proposed change would cause a significant reduction in project planning. It is in this unit that all the topics at the application level are found! Almost all SE programs have a senior

	yearlong capstone project and several other smaller software development projects. It is this section on planning that is quite useful for students – in particular since most of them have never planned for a group outcome before.
comment	5.1 Drop the requirement that curriculum designers and instructors should have deep and broad knowledge of SWEBOK
CCSE Response	We believe that curriculum designers and instructors should be aware of the SWEBOK contents, but the emphasis on required SE knowledge is in the SEEK.
comment	5.1 Drop the reference to CDSP. 5.3 Drop the reference to "widely accepted best-practice." Found the curriculum in case studies that illustrate successful and unsuccessful practices -- that is, in actual experience -- rather than in context-free summarizes marketed as "best practice."
CCSE Response	Guideline 13 has been adjusted to clarify, along the lines the reviewer suggests. The term 'widely accepted best practice' has been kept though.
comment	5.2 Drop the fiction that we are giving the students real-world experience, and accept the consequences of abandoning that fiction. Take Guideline 5 seriously and drop material that the typical software engineering undergrad will not mature into prior to graduation.
CCSE Response	Some adjustment has been made to Guideline 3. It is not clear what other wording changes the reviewer would suggest.
comment	5.3 Reword Curriculum Guideline 15 to push less vehemently the notion that software engineering is a defined profession.
CCSE Response	There has been much discussion and debate about the degree to which SE is or should be considered a “profession”. There have been modifications in chapter 3 to better relate to the emerging nature of SE, and we believe we have struck the proper balance on this issue.
comment	6.3.2 SE221-tes. Fix this description. A broad survey course, which is what you are proposing, is not in-depth. I would recommend instead an in-depth course that covers several aspects of testing but focuses the student either on customer-benefit (e.g. system-level black box) testing or on programmer-focused testing (e.g. unit testing, low-level integration testing, profiling, test-driven development). The course should build skill and experience in the student, preferably with production code.
CCSE Response	There is a new description of the course. All descriptions are now in the appendix. This reviewer may wish to comment in detail, in the final review cycle.
comment	6.4.1 Drop NT291.
CCSE Response	We have kept NT291, but the suggested content has been added.

reviewer	426159 – John Walz
comment	3.2.1 Figure 1. Obscure relationship Either remove Figure or modify by removing red color, adding other connectors besides single double-headed arrow
CCSE response	The figure has been eliminated.
comment	4.15 – Replace ISO/IEEE Standard 12207:requirements of processes with ISO/IEEE 12207 Software Life Cycle Processes: requirements of processes
CCSE response	Modification made.
comment	4.16 – Replace ISO 9000 series with ISO 9000 Quality Management Systems and replace ISO/IEEE Standard 12207 with ISO/IEEE 12207 Software Life Cycle Processes: requirements of processes.
CCSE response	Modification made.
comment	4.16 – Replace “Quality work product attributes include usability, reliability, safety, security, maintainability, flexibility, efficiency, performance and availablility” with “software quality attributes into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability)”
CCSE response	Some modification made. The new quality work product attributes now include all six of the software quality attributes listed above.
comment	4.16 - Replace “tools, and technology” with “tools” since tools are implemented in technology
CCSE response	Noted. The terms in this list should be definable by students.
comment	4.16 – Clarify “The dimensions of quality engineering”.
CCSE response	Noted. This phrase is meant to represent the many facets of quality engineering, not the individual quality attributes.
comment	4.16 – Clarify “Baldridge Award criteria for software engineering”
CCSE response	Modification made. This topic has been changed to “Baldrige Award criteria as applied to software engineering?
comment	4.16 - Add "Root cause analysis and defect prevention" to QUA.pca
CCSE response	For the record, this topic appears as part of product assurance. While it may be true that such analysis can be applied as part of process assurance as well, this would not be a topic that is pertinent to an undergraduate student.

comment	4.16 – Add “Quality process metrics and measurement” to QUA.pca
CCSE response	Noted. Again, this is not a topic that is pertinent to an undergraduate student.

reviewer	2000436873 - Michael Wing
comment	1.2.1 Change “Because computer science provides some of the scientific . . .” to “Because computer science and information science provide the scientific . . .” What other sciences provide any real foundation for computer science – Physics, Zoology The only science worth mentioning (other than computer science) is information science. Other sciences only rear their heads in software applications that support those sciences.
CCSE response	We believe that adding “information science” will only make the confusion about the differences in CS, SE, CE, IS and IT more pronounced.
comment	2.1 [2] Change “value of good engineering design” to “value of good software engineering design” The relationship between software and traditional engineering is still debated, and the principles of engineering can only be applied metaphorically
CCSE response	Agree – recommendation incorporated
comment	2.1 [3] Change “Also, because of the special professional responsibilities of engineers to the public” to “Also, because of the professional responsibilities of software engineers to the public” Many professions (doctors, police, lawyers, engineers, educators) have responsibilities to the public. Engineers and software engineers are not special in this sense. The only thing special is our area of expertise, but every profession has its own special area of expertise. Also, software engineers are not necessarily engineers.
CCSE response	Agree – recommendation incorporated
comment	2.1 [4] Change “changes in technology” to “change in technologies, practices, and applications” Many elements of a profession are not technological. For example, inspections and pair programming are practices, which are independent of technology. Also, application domains (such as web-commerce) can demand entirely new solutions (such as very rapid response) that would otherwise be ignored by traditional technologies and practices. The WWW and E-commerce have forced many practitioners to think and work in new ways. A related change in a later sentence in the paragraph is that institutions must remain abreast of “technologies, practices, and applications.”

CCSE response	Agree – recommendation incorporated.
comment	2.1 [4] Rethink “subject to the constraint of available resources” This clause is trouble. All people and all institutions are subject to the constraints of available resources. In my experience, most problems in software engineering are due to “constraints of available resources.” If educational institutions can excuse anything with “subject to the constraints of available resources”, then Microsoft and the authors of every other buggy, cumbersome application can also excuse their bugs for the same reason. This proposal only uses “subject to the constraint of available resources” in the context of educators, which is a flagrant double standard.
CCSE response	We believe that since CS and SE are such dynamic fields, resources are a major constraint on staying current (more so than in many other disciplines)
comment	2.1 [10] Clarify the meaning of “professional” in this paragraph. Does the adjective mean “as practiced for pay” or “has a license”
CCSE response	The meaning of “professional” is possibly both of these, but much more. The emphasis in this principle is on “profession practice” which is elaborated on in a number of places throughout the volume (e.g., 3.2.1, 3.3, 4.10, Appendix A (NT291))
comment	2.1 [12] Add a principle that “CCSE and educators who teach CCSE must not harm their own students and their abilities to compete in the world economy.” It would be nice for educators to actively foster their own students ability to compete in the world economy (perhaps even to survive the current outsourcing and other economic forces that are destroying their jobs). This is too much to expect, because educators by themselves cannot change the world economy.
CCSE response	Noted.
comment	Alas, I cannot think of a reasonable principle that uses “foster” in a positive way. Fostering is hard for educators, because in the U.S. an engineering degree is supposed to be more general than a technical or vocational degree, which makes goals murky. It is worth noting that in paragraph 2.3 [7], the core should have broad acceptance by the software engineering education community%94 rather than by %93the community of practitioners%94 or %93the whole software engineering community.%94 The (2.3 [7]) principle seems necessary because most professors spend their lives at universities and not on real commercial projects. But, the (2.3 [7]) principle also allows educators to ignore the welfare of practitioners, specifically, their own students a few years later. For example, criticizing practitioners today will harm the students who will become practitioners in a few years. Today%92s students

	are subject to the harm to reputations wrought on previous generations of practitioners by previous generations of academics.
CCSE response	Unfortunately, it would be an unrealistic for the CCSE to get broad acceptance by practitioners, much less the SE education community. However, principle [9] calls for a “broadly based” development of the CCSE, including “industry, commerce and government”. We have worked hard to achieve [9] and believe we have had some success in this area.
comment	2.2 [4] Change “engineering” to “software engineering” to avoid bias.
CCSE response	Agree – recommendation incorporated.
comment	3.1 The effects of software engineering described are very murky. There are hundreds of billions of dollars being spent in the U.S. yearly on software development. Software affects the lives of almost everyone in western economies. Whenever one goes to the bank or uses a credit card, or an ATM, or a car, or television, one uses software. This discussion should be much more accurate. 3.1 According to the 2000 U.S. census, there are 640,000 software engineers in the U.S. and another 530,000 programmers. I do not know what the numbers are for other countries, but there are probably more than 2,000,000 software engineers worldwide.
CCSE response	Agree – recommendation incorporated.
comment	3.1 There is a huge self-definition problem with this section. The authors chose definitions that emphasize ties to traditional engineering, chose to define it in terms of success, and in terms of the Waterfall model. Traditional engineering: There are many places such as http://www.wikipedia.org/wiki/Software_engineering that define software engineering in terms of technologies, practices, applications, and community, without reference to traditional engineering. There are enormous differences between software engineering and traditional engineering. I believe that most software engineers use traditional engineering as a metaphor, for the kinds of success that we want. However, it is not yet clear what the relationship between software engineering and traditional engineering is. Success: Defining any profession in terms of success or failure is problematic. Waterfall Bias: The final paragraph boils software engineering down to the Waterfall model. Side note: The definitions primarily come from IEEE and SEI sources, which appears biased away from the ACM and other

	practitioners.
CCSE response	There has been much debate over this issue and there have been some changes in Chapter 3 that partially address these concerns. Note: The listing of software development activities such as "analysis and evaluation, specification, design, implementation and evolution" should not be interpreted as adopted or advocating a particular software development model.
comment	3.2 This entire section argues that software engineering is a branch of engineering, without acknowledging other possibilities. It also uses the word "professional" ambiguously.
CCSE response	The term "professional" is addressed in other sections and chapters in greater detail.
comment	3.2.1 The comparison in this section applies to nearly every single profession in existence, if you remove the word "engineer."
CCSE response	Although the characteristics might apply to other professions, taken together, we believe they add value to the document. The characteristics have been edited to a more manageable number.
comment	3.2.1 [1] Nearly all human activities are performed in and around buildings. Civil engineers have millennia of head start over software engineering. Most people in 1st world economies use motorize vehicles. Automotive engineers have over 50 years of head start. The second sentence is simply not true.
CCSE response	Characteristic [1] has been eliminated.
comment	3.2.2 The first half of the second paragraph concerns "finding technical solutions to specific practical issues" which also applies to every profession. Judges apply the law to determine whether this person goes to jail or pays a fine. Artists solve technical problems. One "professional" artist from Santa Fe told me that he was trying to use less orange in his sunset landscapes, which is a specific technical problem. The second half of the second paragraph advocates the Waterfall model. The "step-wise approach . . ." is simply the Waterfall by another name. Yet, Royce actually advocated an iterative approach to problem solving in the referenced paper. There are many other approaches to engineering design: Spiral, Agile, and Lean being prominent today. This section could either find true and meaningful comparisons to engineering design, or be rewritten to encompass the breadth of software engineering design. The model describes deployment and operations, which is more of an IT issue. Perhaps this section should explain the relationship between software engineering and IT. (3.2.2)
CCSE	This section has been rewritten. Some of the issues raised by the

response	reviewer have been addressed.
comment	3.3.3 Replace the list “(technical communications, ethics, engineering economics, etc.)” with “(technical communications, ethics, economics, etc.)” Software engineering may need economic principles that are not currently accepted as “engineering economics,” such as Agile and Lean processes. The relative cost of engineering design to production, is very different between software and traditional engineering. The original quote suggests that software engineers will find what they need in existing engineering economics text books and courses. But, there is ample evidence that software engineers need a different body of economic knowledge. Or else Boehm would have simply titled his famous book “More Engineering Economics.” Clarify the word ”professional” in the second paragraph. Does this refer to work in industry, or work with a license.
CCSE response	We believe that the term “engineering economics” is appropriate and certainly would and should include the study of SE economics. The second sentence in the paragraph elaborates on the use of the term “professional” in this context.
comment	3.4 This section primarily discusses the SEI, CSEET, WGSEET, and so on. I know that many individual educators have worked over the past 2 decades at UNM in Albuquerque, thinking about, teaching, and refining software engineering courses. I am personally aware of large efforts by Bruce Wampler and Jeff P. Van Dyke at UNM who taught many students. I have good reason to suspect that thousands of educators around the world have independently struggled to define what software engineering is and how it should be taught. I believe that these individual efforts by thousands of educators drove all progress in software engineering education. Without these enormous efforts, the current CCSE draft would not be possible. The conferences listed in this section were places where hard-working educators presented their triumphs and failures. The current paragraphs suggest that the SEI and other burocracies are taking credit for everyone else’s hard work. There have been many gatherings over the past decades about software engineering education at ICSE and elsewhere. These paragraphs are an opportunity to recognize the pioneering (and mostly thankless) work by these hardworking individuals. Instead, it sounds like political greed by non-participants (mostly the SEI does other things than education). This section could either be expanded to encompass the full

	community of SE educators and the full history of SE education, or it could be moved to an appendix.
CCSE response	Noted.
comment	3.5 The final sentence of the first paragraph is self-contradictory. When a field is “new and dynamic” then keeping the documentation or BOK up-to-date is difficult or impossible, without an enormous effort. When a field is “new and dynamic,” rigid bodies of knowledge should be viewed with great suspicion. When things change rapidly, then the things that one clings to must be small and very true, or one risks getting out of date. The BOKs from all fields should be used carefully.
CCSE response	Noted.
comment	4.1 References to other work should be appropriate. SWEBOK is open to everyone and it is fair to relate it to this proposal. The CSDP is specific to the IEEE. It is appropriate that the IEEE should advocate the CSDP. However, I am a member of the ACM, and I do not want this proposal to imply that I should have joined the IEEE, or that the CSDP is ideal for me or anyone who is not a member of the IEEE. 20 years ago, the ACM had its own certification program that was eventually ignored to death. The CSDP has only 300 people who have passed (by August 2003). At this rate, fewer than 6000 people will pass the exam over the next 20 years. Unless there is a growth pattern that I do not know about, the CDSP is headed for the same fate. This CCSE curriculum proposal should not be limited by a potentially-doomed test.
CCSE Response	The CSDP material was simply referenced as one source. The CCSE effort is not focused on or “limited by” the CSDP.
comment	4.8 Make the Formal construction methods section (CMP.fm.*) optional. This material seems appropriate for an optional upper level or graduate class. These techniques are still mostly in the arena of research, and I have never met a practitioner who actually claimed to use them.
CCSE Response	Noted.
comment	4.9 Remove the engineering bias in this section. Make the title %93Mathematical and Software Engineering Fundamentals.%94 Change the first sentence to %93The mathematical and software engineering fundamentals . . . .%94 Change FND.ef to %93Software engineering foundations.%94 Explain topic FND.ef.6 in terms of the metaphor to engineering. Change FND.ec to %93Software engineering economic.%94 The empirical methods used in software engineering and the economics of software engineering may be very different from traditional

	engineering. Engineering is certainly a metaphor worth exploring. However, claiming that software empirical methods and economics come from traditional engineering implies that students can already learn everything they need to learn from existing engineering textbooks and courses.
CCSE Response	This section was generated by the experts who actively work in this field and added to the SEEK where they felt it belonged.
comment	4.10 Make PRF.pr.1 optional. This material may be required in states and countries, where state licenses exist (like Texas, Canada). However, it is basically meaningless in other states, like California, except for individuals who want to emulate the licensing models in Texas and California. There are many other models of professionalism. I believe that this topic should be very sensitive to everyone outside of Texas and Canada.
CCSE Response	Students should be aware of the current issues facing practicing software engineers.
comment	4.11 Make MAA.md.3 optional. Mathematical models and specification languages remain research topics. Add MAA.md.11 %93Coping with change%94 a required topic. Every project I have ever seen has been affected by change. Skills to cope with change are vital. Make MAA.rsd.3 optional. Specification languages remain research topics. It is unclear when they will become useful enough to be widely adopted.
CCSE Response	Noted.
comment	4.15 Redefine PRO.imp.5 to emphasize team collaboration, rather than team software process. XP and other Agile processes emphasize teamwork, without the measurement and burocracy of the TSP. The TSP is only one possible set of teamwork practices.
CCSE Response	"Team software process" was not meant specifically to be TSP, but to avoid confusion it has been changed to "Team process".
comment	4.16 Make QUA.std and QUA.pro more balanced with actual experience. Clarify that while some companies that adopt these standards have improved, other companies have had seen no improvement. Quality standards and practices are not silver bullets. The %93Lean Software%94 movement defines quality in a similar way to Agile processes. XP has had similar results with ISO-9000 in terms of quality, and should be treated as a quality-oriented process. Also, recognize that the quality standards can have severe negative side effects. Quality processes been used as tools for deskilling jobs, and for exporting jobs from western economies to the developing world.
CCSE Response	These units are here to introduce standards and process only. The amount of actual practice and experience the undergraduate student will have with quality standards and practices will be quite

	limited.
comment	5.1 Remove %93or by being certified in some way (such as the IEEE CSDP certification, or other such designation offered by a professional engineering society).%94 This is a shameless plug for the CSDP. I don%92t know of any other certification programs that are meaningful to this CCSE. Change %93convey to students the engineering-nature%94 to express %93practical,%94 %93real-world,%94 or %93user-oriented%94 nature. The original wording assumes that civil engineers would understand software better than computer scientists, which I doubt.
CCSE Response	The CSDP was included as an “example”, not as the only way to demonstrate knowledge.
comment	5.3 Guideline 7 Remove the engineering bias. Many people disagree that %93software engineering should take its place alongside older branches of engineering.%94 Many people disagree that software engineers should %93embrace the characteristics of engineering.%94 Many disagree that %93students must develop an sense of the engineering ethos.%94 One alternative view is that software engineering is as big and diverse as all of traditional engineering. Another alternative view is that software engineering practices differ substantially from traditional engineering, and that software engineers must find their own way in the world, which is related to, but different from traditional engineering. This guideline should recognize the diversity of opinions about the relationship between software engineering and traditional engineering.
CCSE Response	Guideline 7 has been adjusted, by adding an extra paragraph. This may not satisfy the reviewer, but should clarify that CCSE does not require adherence to a particular political position.
comment	5.3 Guideline 11 Remove the phrase containing %93visual and formal specification languages.%94 These languages remain research tools. It is not clear when they will be widely used.
CCSE Response	Change made, but left “specification languages”.
comment	5.3 Guideline 12 Clarify the %93older tools can be simpler%94 sentence. This logic behind this sentence is muddled. There are many reasons to use older tools, such as compatibility, working on embedded systems that have rudimentary tool support, and maintaining legacy code. Perhaps, the paragraph should argue, %93students should use a wide variety of tools,%94 to better understand when one tool or another would be appropriate, and

	learn how to cope with older and simpler tools.
CCSE Response	Agree – recommendation incorporated.
comment	5.3 Guideline 13 Change %93grounded in sound research and mathematical theory%94 to %93grounded in mathematical theory.%94 The only part of this CCSE proposal that is grounded only in research is the work on formal methods, which has never been widely accepted in practice. Had the guideline required either mathematical theory or practice, then formal methods would be acknowledged as ongoing research material and would be treated as such.
CCSE Response	Other parts of CCSE are also grounded in research, including much work on testing, on user interface design, etc. We feel that the principle is sound.
comment	5.4 Guideline 17 Change %93They should also be exposed to good process and quality so they can experience for themselves the effect of improvement.%94 This is muddled. Students who are exposed to good process and quality may experience pride in their work and learn to appreciate good work.
CCSE Response	Agree – recommendation incorporated.
comment	6.2 Remove the paragraph starting %93Computer science courses in many institutions are taught.%94 This is condescending toward all computer science departments. There may be bad CS teachers, but there will likely be just as many bad SE teachers. Most CS educators I know truly aspire to create better code, but through CS means (which is different than but not worse) than SE means. I don%92t believe that Dijkstra or Knuth would ever tolerate known bad practices. This paragraph is very wrong.
CCSE Response	A valid comment. This paragraph has been rewritten to account for the points raised.
comment	6.2.1 Change %93(2) the engineering perspective%94 to %93(2) the software engineering perspective.%94 The previous working implies that software engineering is a branch of traditional engineering.
CCSE Response	Agree - the wording was adjusted so that the comment about engineering perspective applies to SE101/102 only.
comment	6.3.2 SE312-lld Remove %93including formal approaches%94 That is what compilers are for. Formal approaches remain research tools.
CCSE Response	This path through the program is more suitable for those who believe that software engineers should be taught formal approaches, and that they are more than research tools. Some universities want this, and many believe this is entirely appropriate.
comment	6.3.2 SE313-fm Make this class an optional topics course for

	seniors and graduate students. This topic remains a research topic.
CCSE Response	Others argue the opposite. We believe we have struck a balance. In particular, the course sequences can allow for considerable flexibility in the amount of coverage of formal methods.
comment	6.4.1 NT271-eco Change the name to %93Software Engineering Economics.%94 The relationship between software engineering and traditional engineering remains fuzzy. NT291-eth Remove this course, or make it optional. I looked around on the web, and I did not find any engineering departments that have a %93professionalism%94 course. On the whole this course is murky. It appears to be one more opportunity to brainwash students. This course may be appropriate in those jurisdictions where software engineering is licensed, such as Canada and Texas, but it is meaningless or condescending elsewhere.
CCSE Response	Noted. NT291-eth covers a lot of material that people feel is quite important.
comment	6.4.2 The discussion about the role of calculus is telling. Students take only a limited number of courses. Devoting effort to calculus means that they cannot talk other courses. It would be nice for every student to take 20 years of college, but that is unrealistic. This curriculum must define the priorities for software engineering students. The relative unimportance of calculus distinguishes software engineering from all other branches of engineering. The sentence starting %93Other mathematics commonly%94 might point out that these courses also support numerical analysis and modeling for the sciences (like physics). The sentence %93Taking some science and engineering courses will . .. .%94 is also telling. This also has to do with priorities for all software engineering students. If all software engineers would write programs for scientific and engineering modeling, then requiring science and engineering courses would make sense. Instead, because of the limited hours in a 4 year degree, students should take classes relevant to the domains they will work in. This difference in priorities also separates software engineering from traditional engineering.
CCSE Response	The Calculus is not a required SEEK topic, but as discussed in 6.4.2, there are some programs that will want to require it.

reviewer	4614071 – Don Bagert

comment	4.14 – Too little coverage of evolution; either increase the hours or remove this KA and distribute the topics within other Kas.
CCSE response	Noted. Evolution is a difficult topic (as well as process) to teach at the undergraduate level; particularly since these students have had so little experience. It is for this reasons that you give as to the importance/significance of evolution that the we strongly believe that EVO should remain as a KA. The number of hours listed, as well as the topics covered, was suggested by a renowned expert in software maintenance and we support his recommendation.
comment	6.3 Add a third core software engineering package which does not include a separate HCI course. One model would be based on the six course sequence in the 1989 SEI graduate curriculum model, which was adapted by Rose-Hulman for their undergraduate core, and subsequently mapped to SEEK by Mark Ardis (the co-creator of the SEI model). Mark will be presenting these results as part of an FIE 2003 panel.
CCSE Response	It might be possible to distribute the HCI material among other courses, but this would result in somewhat unfocused mixtures. Given the recognition (perhaps late) of the high importance of user interfaces, a required course seems appropriate. The SEI curriculum dates back to 1989. That is a long time ago; we can make progress and realize that improvements can be made.
comment	6.3.1 I suggest that a class called "Software Design and Evolution" is ill-matched. Either evolution issues should be discussed throughout the core sequence (which would mean that the title of the course should be changed) or evolution should be coupled with construction.
CCSE Response	The name has been changed to Software Design and Architecture. However, the material has not been moved to the construction section since it is not clear why this would be better. The concepts of evolution apply at all scales, but become most obvious when designing architectures (which must permit evolution). It is true that maintenance tends to be done mostly at the low (construction) level, but the actual material that needs “teaching” is mostly higher level in nature.

reviewer	4714197 – Volodymyr Pavlov
comment	All references to CMM/CMMI should also reference the Organizational Project Management Maturity Model (OPM3) since this will be the “standard” by the time that CCSE is released
CCSE response	At this time, the only reference to CMMI is in software quality processes. It is not clear how this topic relates to the Organizational Project Management Maturity Model. Once this indeed becomes the standard and it is clear where such a reference belongs, future releases of the CCSE will contain the

	reference.

reviewer	20004816967 - Wes Doonan
comment	A comprehensive treatment of the subject matter; clearly much good work has gone into achieving this level of maturity in the document. Incorporation of practical learning with theoretical background highly appreciated. Focus on software construction as an engineering discipline is appreciated (and sorely needed). A general bias towards the design and coding aspects of complex software construction still permeates the document. Clearly some effort has been made to encompass the other important areas of the development process, this effort should be encouraged and intensified.
CCSE response	Noted.
comment	3.2.1 item 7 - This point could perhaps be expanded to also recognize the generally immature nature of tools available to software engineers, and the critical need for the design and development of better tools for use in constructing complex software systems.
CCSE Response	The relationship between tools and an SE curriculum are addressed in a later chapter.
comment	4.8 Add some discussion of resource limitations of programs, consequences thereof. That is, memory size limitations, compute cycle limits, the effect these limits have of design choices. Correct program construction when faced with real-world limitations of such resources; including limits on network bandwidth.
CCSE Response	At one time, we had a separate topic in CMP on just such an issue but it has been incorporated into DES under design tradeoffs.
comment	4.18 – I believe that the development of tools for furthering the practice of software engineering itself is an area at least equally important to other application areas explicitly identified (or alluded to) in the document. I also think this area is woefully under-served, and should be presented to undergraduates as an important area for investigation and study.
CCSE Response	The list of specialty areas is in no way meant to be exhaustive.
comment	FND.mf.10 - Even a survey course in number theory would be of real use, particularly in preparing students to appreciate the concepts that underlie security architectures.
CCSE Response	Noted. It is true that this topic would be of great benefit to programs that security. In particular, such a topic would be essential as part of a SSA in secure systems (sec).

comment	5.2, item 5 - This point is quite important, and should be impressed upon curriculum designers as such. Curriculum designers might be encouraged to build tracks where students are organized to work in teams on complex projects quite early in their academic career, so that they are immediately exposed to both the human and technical challenges involved in teamwork.
CCSE Response	We have enhanced the text concerning this point.
comment	5.2 item 6 - I fundamentally agree that SE students should have exposure to non-SE areas as part of the curriculum. However I am biased against curricula that would create "software for telecom" or "software for databases" or other such courses. As electives, possibly, but not in core.
CCSE Response	The guideline does not suggest these have to be core courses; and there are no such core courses among the courses we propose.
comment	5.2 item 14 - Project-based courses are a great idea, and should include having the student serve in each of the many roles present in software development teams -- not just as designers. Students should serve as a verification engineer, a tools engineer, a project manager, a requirements engineer, a systems engineer -- each of these roles is typically present in a software project team, each should be experienced.
CCSE Response	Agree – recommendation incorporated.
comment	6.2 - The important point I think is not whether there are more CS or SE courses early in the curriculum, but rather that the principles of engineering as a problem solving activity (and hence inculcating the analysis and design skills necessary to become proficient problem solvers) be taught right at the beginning, so that other foundation courses can be interpreted in the correct light (e.g. so that data structures and algorithms are viewed as a means to an end, tools for solving problems, not necessarily ends to themselves).
CCSE Response	The text has been adjusted to make this clearer.
comment	6.3.3 - Including a capstone project is a great idea. It would seem wise however not to characterize it as a "significant design experience" so much as a "significant development/delivery experience". The former gives the traditional bias towards creation/coding, possibly giving short shrift to the other equally important aspects of the overall process (requirements, management, quality, verification, metrics, feedback, etc).
CCSE Response	Agree – recommendation incorporated.
comment	6.4.1 NTxx courses - Hooray these are terrific ideas, should

	definitely be part of all SE curricula somewhere.
CCSE Response	Noted.
comment	6.4.1 - Increasingly software engineers must be aware of the mechanics of the sales and marketing process. What a customer is, how they are motivated, how best to interact with them, etc. Also how customer support works after the sale. Somehow exposing students to material of this sort would be very beneficial, again in inculcating the notion of software as a product, serving a need and a user, solving a problem, all of which underpin software engineering as Engineering rather than Science.
CCSE Response	Adding a course in marketing is a good idea, but would probably not be reasonable as a recommendation for every single program. On the other hand, we are suggesting that course NT181 include quite a bit of what you propose.

reviewer	20004922095 - Ivan Mistrik
comment	Omission Using Design Rules (Standards & Guidelines to direct design activity)
CCSE Response	This is included in “DES.con Design concepts” in the SEEK.
comment	Omission Software Engineering Life Cycle (distinct activities and consequences for interactive system design)
CCSE Response	This is included in “PRO.imp.2 Life cycle models”, in the SEEK.
comment	Omission Iterative Design & Prototyping (limited functionality simulations and animations)
CCSE Response	This is included in “DES.con Design concepts” and “DES.str Design strategies”, in the SEEK.
comment	Omission Design Rationale (recording design knowledge process vs. structure)
CCSE Response	This is included in “DES.con Design concepts”, in the SEEK.
comment	Omission Usability Engineering (making usability measurements explicit as requirements)
CCSE Response	This is included in “MAA.af.3Analyzing quality (non-functional) requirements” and “DES.con.6 Design for quality attributes” , in the SEEK.
comment	Note: The reviewer made a number of suggestions about changes to the SEEK topics associated with SE courses, listed in Appendix A.

CCSE Response	These changes would first have to be done in SEEK. SEEK coverage in the appendix is semi-automatically generated. Also, material on “Learning Objectives” and “Additional considerations” have been added.

reviewer	20005029840 - Pete Knoke
comment	The organization of CCSE is excellent, and the content is mainly very good. It was also interesting to read. I think the use of color in some chapters helped with readability, and showed patterns in what could be a useful way (but I'm still trying to figure out the possible meaning of some of the patterns). What is there is very comprehensive (i.e., Chapters 1-7, and Appendices A,B, and C). Also, I think the Bibliography for Software Engineering Education is quite good.
CCSE response	Noted.
comment	Note: The reviewer pointed out numerous errors in grammar and style.
CCSE Response	Errors corrected.

reviewer	200051157 - Michael Lutz
comment	4.15 Drop PRO.imp.4 and PRO.imp.5 from application (a) to comprehension (c) in the Bloom level.
CCSE Response	Agree – recommendation incorporated.
comment	4.18 SAS.sfy - formal methods and control systems are not sufficient. Students need exposure to Failure Modes and Effects Analysis and Fault Tree Analysis. Indeed, I'd weight these above formal methods if push came to shove.
CCSE Response	Agree – recommendation incorporated.
comment	6.3.2 - Reverse the prerequisite order for SE213 and SE 312.
CCSE Response	This sequence was designed to allow for coverage of large-scale architectures before coverage of lower-level design issues ... as in a top-down approach. Some see this as appropriate. For those who do not like it, then Core Software Engineerinng Pacakge I would be better.
comment	Appendix A - SE312 needs to have some coverage of refactoring. SE312 shows low cohesion. It addresses issues ranging from construction tactics through basic design up to and including formal modeling.
CCSE Response	We have added refactoring. We don not believe, however, that including formal approaches is a problem; many people believe that a certain amount of formality should be encouraged in each

	design course ... and that such material should not be relegated exclusively to a separate course. Note that this course is part of pattern 2; curriculum designers who don't like this way of packaging material can pick pattern 1.
comment	Appendix A - SE313 is a pastiche of topics that do not, on the whole, fit well together.
CCSE Response	While some people would cover one or another of the areas in this course in great depth, we have created a course that covers many of the formal topics from SEEK in moderate depth. We believe that such a course would expose students to the variety of types of formalism.

reviewer	2000523841 - David Rine
comment	The following historical background paragraph should be inserted in section 3.4. Thank you. In 1976 The IEEE Computer Society activated is Education Area, under the supervision of Dr. David Rine, to undertake a first draft development of a "Model Curriculum for A Masters Degree in Software Engineering ". The publication was co-edited by David Rine, Richard Fairley and Anthony Wasserman. The Outcome of this Education Area work was a publication MSE-78: Model Curriculum for a Masters Degree in Software Engineering …
CCSE response	Reference to this work has been included in Section 3.4.

reviewer	20005325573 - Bruce H. Barnes
comment	I like the report very much. The group has made considerable progress and I am looking forward to seeing the complete report. I especially liked the way that they started with requirements, CCSE Principles, and developed specifications, Curriculum Outcomes and SE knowledge areas, before moving on to implementation. This is a very logical and appropriate approach for a group of software engineers. The Pedagogical group should complete the specifications and move into the design phase. I did not see that there are plans for producing one or more sample curricula. This is imperative! Computer Science is a well-established academic major. There are hundreds of ongoing programs graduating a considerable number of graduates. The broader community, i. e. faculty, administrators, students, employers and parents, has a relatively good vision of what such programs entail. This is not true for Software Engineering where there are just a handful of established programs. The report must not only be a set of

	guidelines for a set of knowledgeable faculty to develop a software engineering program or modify an existing one, it needs to be a spokesperson for the discipline. It is necessary for faculty, administrators and employers to be able to visualize a complete program in an academic setting.
CCSE Response	Detail has been added to the course specifications in Appendix A (learning objectives, additional considerations). Also, additional curriculum patterns have been added, which serve as "sample curricula".
comment	Omission - There are two implied requirements, i.e. principles, that should be included. These are 1) that the report is to be consistent with accreditation guidelines and 2) be built upon the recommendation in CC2001. These do not change anything, but they give creditability to the report and answer a lot of questions before they are asked.
CCSE Response	1) Principle 3 in Section 2.1 includes that statement " … it is important that the curriculum guidance support and promote effective external assessment and accreditation of software engineering programs. Also, we have added a section "8.2 Assessment and Accreditation Issues" in a new Chapter 8: Program Implementation and Assessment 2) The statement "The following list of principles were strongly influenced by the principles set down in the CCCS volume" precedes the principles in 2.1. In addition, in numerous places throughout the CCSE volume reference is made to the influence of the CCCS volume.
comment	Omission - Computer and network security plays an essential role in most commercial and military applications and deserves more than one hour in the core requirements. The topic depth in security, as found in some of the specialties, should be included in the core.
CCSE Response	While security issues have been increasing in importance, little is found in current undergraduate curricula on it (which most likely explains why little to no reference has been found in other KAs). As the exposure to security issues increases in the classrooms, it will also be found in more KAs.