Review Comments for SE2004 Volume - Public Draft 3 - 1/25/04

This document represents the response to reviewer comments from the second and third public calls for review of the SE2004 Volume (previously named CCSE). The second public review was completed in February 2004, and third in March 2004. Each comment is paired with a response form the SE2004 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. The SE2004 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	3000027837 - Hareton Leung (Hong Kong Polytechnic University)
comment	5.4 For curriculum guideline 17, it may be better to mention cyber-based learning.
CCSE response	Good point. We have made this explicit.
comment	5.4 Include a guideline on how often the curriculum should be reviewed to incorporate the latest development in SE.
CCSE response	CCSE principles 3 and 4, in section 2.1 address the need for CCSE itself to be kept up to date, but the reviewer is correct in stating that the principles for individual curricula in Chapter 5 do not also make this point. We have therefore added a guideline 19 that explicitly makes this point

reviewer	30000313617 - Tony Cowling (University of Sheffield)
comment	2.2 As far as I can tell, this was originally supposed to be a separate chapter between the current chapters 3 and 4. This really where it belongs, and it ought to be moved there.
CCSE response	We agree that a context about the nature of the SE discipline should be laid prior to espousing guidelines and outcomes. Chapters 2 and 3 have been interchanged.
comment	3.2 Modify the wording of the fourth sentence - seems to be claiming that the SE-CS connection is stronger than the Eng-Sci connection for other branches of Eng, which is probably not true, whereas I think that what it is trying to say is that SE is more strongly connected to CS than to other branches of Eng, which seems much more believable.
CCSE response	We agree and have incorporated your recommended change.
comment	3.2.1 Reinstate the following characteristic that was listed in public draft 1, and has now been lost: that engineers solve customer problems. The customer focus is one of the key things that distinguish engineering from science.

	3.2.1 Reinstate the following characteristic that was listed in public draft 1, and has now been lost: that engineering is a creative discipline. While item 3 refers to "creating a design", the rest of these characteristics make it all seem very mechanical, and it isn't!
	3.2.1 Reinstate the following characteristic that was listed in public draft 1, and has now been lost: that engineers have to apply knowledge from other disciplines. Engineering can not be seen as an activity that stands in isolation from other disciplines.
	3.2.1 Reinstate the following characteristic that was listed in public draft 1, and has now been lost: that engineers work in teams (often interdisciplinary). Because the lone engineer is almost a contradiction in terms!
	3.2.1 Possibly make a distinction between those characteristics in this list that are specific to engineering (e.g. the development of systems to solve customer requirements), and those that are shared with other professions. Otherwise, it looks as if more is being claimed as exclusive to engineering than is reasonable.
	3.2.1 Either add to the items in this list comments on how they apply to SE, or possibly follow this list with a separate matching list of points that discuss the application of these characteristics to SE. Some of this was there in public version 1, and seems to have been lost.
CCSE response	Those items removed were characteristics of more than just all engineering disciplines, they included most craft/licensed professions. An attempt was made to restrict this list to those characteristics that were particular to engineers. However, because of its importance and in SE and in SE curriculum design, the teamwork characteristic will be added back into the list of characteristics.
comment	3.3.1 Currently this refers to a number of the points that were in 3.2.1, and have now been lost from it. If the proposals above to reinstate them are not accepted, then references here to those points need to be removed - but I would prefer to see the points reinstated in 3.2.1.
CCSE response	This section has been changed to accommodate the changes in the section characteristics of engineering.
comment	3.5 Since the bullet points here have been re-ordered from the previous version, the citation of [Bloom 1956] needs to move up from the third bullet point to the first one.
CCSE response	The Bloom citation is included in the correct bullet item.
comment	4.1 To acknowledge, at the end of para 2, that actually students can achieve higher levels on the Bloom taxonomy than the three that are used in the SEEK, and the restriction to 3 levels was really to keep the classification manageable. Our students do - perhaps

	not as frequently as I'd like, but often enough that the current statement is just not true.
CCSE response	The steering committee selected the first three levels of learning given that they represent what undergraduate programs, across the international spectrum, should be expected to achieve. Some programs, in some areas, such as yours, will set and achieve higher expectations.
comment	4.3 Add, to the explanation of the limits of the core, the observation that was made at the SIGCSE meeting, viz that the reason why some KAs had comparatively few core hours was because it was envisaged that students would learn most of what they needed in these areas in professional practice after graduation.
CCSE response	It has been stated repeatedly throughout this document that the SEEK is limited to what can be learned in an undergraduate SE program. In particular, this distinction is made when referencing SWEBOK and the differences between the two documents.
comment	4.9 Delete the note at the end about topics 1-6 corresponding to CCCS DS, since this is now in the entries in the table.
CCSE response	It is indeed true that the individual DS topics are referenced on some of the topics of FND.mf. The footnote explains what the CCCS DS reference means and we have decided to keep it.
comment	4.11 Consider splitting MAA.tm.2 into two topics, so as to separate reliability modeling (failure modes and effects analysis, fault tree analysis, etc) from the more basic behavioral modeling (the rest). Because typically the basic material is introduced some time before the reliability modelling, and separating them would make this clearer.
CCSE response	This section can be separated in two different categories of modeling but they are presented together since both categories represent types of models. As stated in the beginning sections of chapter 4, the ordering of KAs, Units, or Topics is not meant to represent the order in which the material is to be taught.
comment	4.11Remove the self-references under "related topics" in MAA.er.2 and MAA.rv.1. Because such self-references don't make sense!
CCSE response	You are correct. MAA.er.2 should reference MAA.er.1. MAA.rv.1 should not reference itself. The errors have been corrected.
comment	4.18 In SAS.sfy.3, add at the start of the wording "Depth in". The basics are already covered in MAA.tm.2.
CCSE response	The recommended wording has been added.

reviewer	30000515713 - Michael Wing (Vandyke Software)
comment	Positive Comments The CCSE curriculum proposal draft 3.1 (specifically the courses)

	is very good. The courses introduce all of the important software engineering material that I know. I am impressed with its sophistication and respect for diverse opinions. For example, the discussion of licensing respects the disparate opinions of people and properly defers resolution until later, when a consensus may emerge. I would quibble that the curriculum still has too much process and formal methods and not enough construction and testing. I looked up the course offerings of mechanical engineering at MIT and electrical engineering at Stanford, and I note that they spend much less time on process. But I can live with the current proposal.
CCSE response	We have attempted to reach an appropriate and reasonable balance among such topics as process, formal methods, construction, and testing. We have purposely left sufficient flexibility in the recommendations in order to support diversity and specialization among BSSE programs.
comment	Negative Comments However, I protest the assertion that software engineering is a branch of traditional engineering. Software Engineering Stands on Its Own According to the statistics presented in Software Engineering is Very Big, software engineering is about 60% as large as all traditional engineering combined, and with a small change of perspective, it may soon grow to be as large as all traditional engineering combined. In the U.S., software engineering drives an economic sector that earns \$200 billion to \$250 billion every year and drove \$1 trillion of economic growth over the last decade. Software engineering is evolving rapidly, and it makes a positive difference in the lives of more than more than a billion people, every day. Software engineering is big, important, very cool, and is emerging to stand on its own. Software engineering is worth fighting for. The claim that software engineering is a branch of traditional engineering has many negative consequences, both inside this curriculum proposal and in the real world. Inside this proposal, the claim affects the rhetoric of the proposal; whether students must study years of calculus, physics, and chemistry; what kind of math is appropriate; and which departments are qualified to teach it. In the real world, this claim affects jobs; recognition for leaders, educators, and researchers; the role of the ACM in software engineering; and the future status of software engineering as a profession. This claim affects us all. The CCSE proposal deliberately downplays the size and

	importance of SE, so they make the claim that SE is a branch of engineering seems unimportant. But in doing so, they also downplay the size and importance of the SE profession and all SE practitioners, which also affects us all. I understand why traditional engineering wants to steal a profession as cool and important as software engineering. Now is the time to steal it, before software engineering finds its own identity and stands up as its own profession on its own terms. But, it is job of computer scientists and of the ACM to defend their contributions to software engineering, the legacy that software engineering inherits, and their right to participate in the future of SE. The specific relationship between traditional engineering and software engineering should be removed from this document, until after a consensus emerges. The relationship is not needed to justify any of the other core material or proposed courses. Of course, it is reasonable to state that within Canada and Texas, software engineering is a branch of traditional engineering by legal status. However such a claim has not been made in New Mexico, California, New York, Japan, Israel, or in most other jurisdictions. Note that if software engineering is a branch of traditional engineering, then it is the biggest branch by a long shot. It is so large that it can write its own rules. Note that the concluding paragraph of 3.2.1 on page 16, states that software engineering is beyond engineering.
CCSE response	There has been a great deal of comment and discussion about the description of SE in chapter 3 (see review comments for the first public review). We have reworded sections to improve clarity and, in some places, modified the tone and emphasis. Some have criticized the emphasis on the “engineering” nature of software development and others have called for drawing a stronger relationship. We have attempted a complex balancing act of both reflecting the current state of SE and trying to project what its future should be (for purposes of educating the SEs of the next generation). We appreciate and learn from constructive criticism. The statement “… CCSE proposal deliberately downplays the size and importance of SE…” is untrue and unfortunate.
comment	Calculus, Physics, and Chemistry The proposal clearly states that most students will find calculus, physics, and chemistry useless. Yet, it advocates wasting many hours of course work for all students to follow in the footsteps of other engineers. There is probably better material for them to

	learn. Kind of Mathematics The proposal advocates more mathematics, which I agree with. But, calculus is only one kind of mathematics. The proposal also disparages theoretical computer science, which is another kind of mathematics. I believe that any kind of mathematics is fine for teaching mathematical rigor, but the proposal clearly prefers (the useless) calculus over (the possibly useful) theoretical computer science. Note that section 1.3 clearly states that CS provides many of the underpinnings of SE.
CCSE response	Calculus, physics and chemistry are <u>not</u> part of the core SEEK areas that are recommended for all SE programs. Some programs however, will want to include such courses (e.g., they specialize in domains that require such knowledge, they are part of a college of engineering that requires continuous math and science courses, or for other historical reasons). The mathematical topics that are part of the core come from discrete mathematics and mathematical logic. There is no intent to disparage theoretical computer science; in fact, we believe that a foundation in computer science theory is essential to the study of software engineering. That is why the largest percentage of contact hours in the SEEK is devoted to computing Essentials. However, we would expect that a greater percentage of a CS curriculum would be devoted to purely theoretical topics than in SE, since the objective of many CS programs is prepare students for advance study and research in computer science.
comment	Engineering Mindset Section 5.2 Guideline 4: states that students must develop an “engineering mindset.” What does “engineering mindset” really mean. All of my colleagues who have CS degrees understand measurement, modeling, and abstraction, problem solving, and reuse just fine. Many non-engineers I know also understand these concepts. This guideline reminds me of the “splunge” skit from Monty Python should be removed. This guideline implies that few of the educators on the CCSE committee have an “engineering mindset” and are unqualified to work on this proposal. Edsger Dijkstra has 3 degrees in physics, which means that he would have lacked the mindset to contribute to either computer science or software engineering. Barry Boehm has 3 degrees in mathematics, which would mean that he has always lacked the “engineering mindset” to contribute to software engineering. David Parnas has math and CS degrees, and so lacks the proper mindset. Clearly, this guideline is silly.

	Besides, the report by the Sloan Foundation http://www.cpst.org/S&E.pdf compares engineering and computer science mentalities, and finds that they are almost identical.
CCSE response	Guideline 4 discusses how a curriculum should use recurring themes to help students develop a "software engineering mindset". The emphasis is on the listed topic areas that should recur throughout a curriculum. It is true that many mathematicians and computer scientists also have many of the mental abilities and skills that constitute a software engineering mindset. So Parnas, Dijkstra, and others the reviewer mentions, first of all are very sharp minds, founders of the field, and of course, are fine software engineers. However, there are plenty of people developing computer systems who don't think in terms of measurement, modeling and abstraction, etc. We want to ensure future software engineers do.
comment	2.1 The phrase "good engineering design" should be "good software engineering design" to recognize the many non-engineering sources of design inspiration. Patterns come from architecture.
CCSE response	Sorry, we disagree. First, the context of the use of the term "good engineering design" in this paragraph is clearly identified as in the field of software engineering. Second, the statement is meant to appeal to the tradition and experience of the engineering disciplines in their emphasis on design in the broadest sense of effective product development.
comment	2.1 CCSE Principles The term "special professional responsibilities of software engineers to the public" ignores the fact that all professions have responsibilities to the public. Which responsibilities are special is unspecified. I doubt that they exist. The need for curriculum guidance and assessment and accreditation is necessary for all professions.
CCSE response	We agree that such statements could be made about other professions and they typically "are" made in documents such as the CCSE volume, in order to emphasize their importance. The ACM and IEEE-CS have described such "responsibilities" in the "Software Engineering Code of Ethics and Professional Practice".
comment	2.1 The phrase "subject to the constraints of available resources" is a problematic double standard, which allows educators to ignore their shortcomings away, while practitioners are not given such an out for their efforts.
CCSE response	The statement is related to the principle of being *"sensitive to changes in technologies, practices, and applications, new*

	developments in pedagogy…" Just as software projects have constraints on resources, schedule, technology, etc., academic institutions have constraints: they do not have unlimited budgets for new technology, etc. We certainly hope they have sufficient resources to keep their programs current.
comment	2.2 The phrase "deliver quality software artifacts" should be changed to "deliver appropriate software artifacts" to reflect that engineers deliver all kinds of artifacts. They also make prototypes, proofs of concepts, and so on, where quality is not the right adjective.
CCSE response	Not sure we understand the reviewers point. For example, a prototype may not be defect free, but for its purposes (e.g., help elicit customer requirements), one would want it to be of high quality.
comment	3.1 There are hundreds of billions of dollars spent of software engineering every year, and more than 600,000 practicing software engineers in the U.S. Using small numbers serves to minimize the value of software engineering. If the numbers refer to software engineers in Canada or Texas, please state the context.
CCSE response	We have changed the numbers to accord with your data.
comment	3.1 Please find definitions of software engineering that are more tolerant of the diverse opinions. These definitions are biased towards the IEEE and SEI. They limit the vision of what software engineering could be. The first 2 definitions emphasize traditional engineering, which is only one possibility, and ignores the many definitions that need not be biased by such a connection. The Wikipedia entry http://en.wikipedia.org/wiki/Software_engineering defines software engineering in terms of technologies, practices, applications, and community, and so it applies for those who aspire to traditional engineering as well as those who would seek other models for the profession. The 3rd definition emphasizes "systematic" and "quantifiable" which are classic "scientific management" terms that are closer to process than to engineering. I am unaware of any important definition of chemical engineering that uses those terms. Many modern concepts of software engineering (agile processes and open source development) cannot be understood in those terms.
CCSE response	There has been a great deal of comment and discussion about the definition of software engineering. We have chosen definitions that are widely used, represent some collective opinion (the Bauer

	definition was included partially for historical reasons), and are helpful in the design of an undergraduate curriculum. Although the Wikipedia definition of SE ("Software engineering (SE) is the profession concerned with creating and maintaining software applications by applying computer science (CS)") has merit, we feel it would not provide significant added value to the definitions used in the document.
comment	3.1 The phrase "in the engineering tradition" focuses attention on engineering, when almost all fields draw upon a broad range of disciplines.
CCSE response	It is true that other disciplines draw on a broad range of other disciplines. However, we felt, for the purpose of this document, it was important to emphasize this for engineering in general and SE in particular: " … software engineering builds on computer science and mathematics" and "it goes beyond this technical basis to draw upon a broader range of disciplines".
comment	3.1 The phrase "high-quality software in a systematic, controlled, and efficient manner" is problematic. Using the phrase "appropriate manner" would recognize that software projects have all sorts of characteristics, and the ideals vary from project to project.
CCSE response	We believe "systematic, controlled, and efficient" are good adjectives and are appropriate goals for SE practice. Certainly it is possible to produce high-quality software in an unsystematic, uncontrolled and inefficient manner, but the business objectives for the product probably would not be met.
comment	3.2 The implication "Thus, the discipline can be seen as an engineering field" is spurious. It can also be seen as many other kinds of fields, which would be mentioned here, were the bias towards them. Please be more broad minded about SE.
CCSE response	Since SE has emerged as a discipline in the last 20 or 30 years, there has been extensive debate and discussion about to what degree SE is an engineering discipline. We believe that although SE is not fully formed, it has matured in many ways. The purpose of this section was to point to the similarities and the differences between SE and the established engineering disciplines.
comment	3.2 The list of differences should include the economics of projects (traditional engineering projects are usually overwhelmed by construction and manufacturing costs), constraints of manufacturing and construction on change (software changes can be much faster than rebuilding a house or road), and the very long history that buried many of the engineering problems in the mists of time so we forget about them.
CSE response	These differences have been incorporated in the list.
comment	3.2.1 I know of many examples from chemical engineering, where

	engineers do not follow strict processes. Engineers often work in a very unstructured way as firefighters to make the chemical plants and construction projects succeed.
CCSE response	Noted.
comment	3.2.1 The final paragraph is correctly generous to CS and other fields that inspired software engineering. Expand on this.
CCSE response	There are numerous places throughout the document where the value of CS and other disciplines to SE are noted (especially CS). This is emphasized in the guidelines, the SEEK, the course descriptions, the curriculum patterns, etc. We have reinforced this in several other parts of the document.
comment	3.2.2 I am not aware of any major body of work known as “engineering design” that describes what software engineers do. Many other fields do design, too. Software engineers have drawn from many sources (patterns come from architecture), and so “software engineering design” should recognize the scope of our sources. To obsess over engineering design is to deliberately ignore broader perspectives, which may for example be inspired by the domain that the practitioner works in.
CCSE response	We believe that design is a central and crucial activity of software engineering (sections 4.11 and 4.12 address the core modeling and design knowledge specified for undergraduates), and that this section properly discusses the similarities and differences between traditional engineering design and software engineering design.
comment	3.2.3 This section is too biased towards traditional engineers.
CCSE response	We believe this section is relevant to issues in software engineering practice (domain engineering, component-based development, build vs buy, product-line development, etc.).
comment	3.3.1 Many professions have obligations to the public. Please state what the obligations are, so that we may know whether they are engineering obligations or not. The list presented in the paragraph applies to almost all fields, independent of engineering.
CCSE response	The next section of the document discusses the *Software Engineering Code of Ethics and Professional Practice,* which has additional detail about an SE’s obligations to the public. It is true that the list does apply to many fields (in addition to software engineering); however, in providing guidance to educators, we felt that it was important to emphasize this area (as has been done in ACM CS curriculum guidelines for decades).
comment	3.3.1 The last sentence “software engineers need to seek quantitative data” covers all bases in the murkiest way possible. Perhaps it should read something like “Like everyone else, software engineers seek to make appropriate decisions, using appropriate information, which when appropriate may include appropriate quantitative data.” The previous version emphasizes

	scientific management and contradicts agile and lean processes.
CCSE response	We agree that the statement may be a bit too rigid and have edited it to provide more flexibility and judgment about the use of quantitative data in decision making
comment	4.7 Change “Mathematical and Engineering Fundamentals” to “Mathematical and Software Engineering Fundamentals”, to minimize bias towards traditional engineering. Discuss the “Software engineering” foundations and economics.
CCSE response	The title was chosen from our list of our foundations: namely, computer science, mathematics, and engineering.
comment	4.7 Remove the sentences about continuous math and natural sciences. This implies that software engineering will be taught in engineering departments.
CCSE response	Continuous math and the natural sciences are not part of the core SEEK areas that are recommended for all SE programs. Some programs however, will want to include such courses (e.g., they specialize in domains that require such knowledge, they are part of a college of engineering that requires continuous math and science courses, or for other historical reasons) There are many CS programs, which are not part of engineering departments or colleges, that require calculus and natural science courses.
comment	4.8 Make “Formal Construction Methods” section optional. These remain areas of research, but have not been proven in practice, except under special circumstances. When these are proven, they should become part of the curriculum.
CCSE response	The level to which such material is recommended is appropriate for a topic that appears in a significant number of software development textbooks. Teaching and experiencing rigorous software development enhances a student’s understanding of the meaning of their programs independent of whether they may be asked to apply directly such techniques in the workforce.
comment	4.9 Re-title everything in this section as “Software Engineering Fundamentals” to avoid bias toward traditional engineering, and recognize scope of software engineering. Similarly, cite “software engineering design” and “software engineering sciences”
CCSE response	The sections that follow concentrate on software engineering, including section 4.12 devoted to software design. Sections 4.8 and 4.9 describe areas that influence software engineering knowledge (CS, math, and general engineering).
comment	4.9 The section FND.ef.3 has been shown to be very counterproductive in practice. People respond much better to peer pressure rather than to metrics of individual performance. This principle show archaic “scientific management” practices, rather than contemporary practices based on teamwork.
CCSE response	Good point. The emphasis is meant to be on the use of metrics in making decisions and assessment, not for management to use

	metrics to evaluate and an individual's performance. FND.ef.3 has been changed to "Measurement and metrics",
comment	4.10 The sections PRF.pr.1, 4, and 5 contradict the statement of professionalism in section 3. The topics of "licensing" and so on depend on the jurisdiction, and what is taught should be clarified. This appears to require that licensing and the IEEE must be taught.
CCSE response	As part of the professionalism of SE, it is reasonable to inform students of the existence of our professional societies (ACM as well as IEEE) and the role in which they play. We also believe it is reasonable to discuss issues surrounding accreditation, certification, and licensing since they directly affect the SE.
comment	4.11 Modeling and analysis are core concepts of many disciplines, including marketing, sales, movies, etc.
CCSE response	This may be true; but the modeling and analysis detailed in this KA correspond to an SE.
comment	4.11 In line MAA.tm.8, non-functional requirements are rarely defined in terms of quality. They are "non-functional" requirements.
CCSE response	We do not understand the comment. MAA.tm.8 is not specifically about non-functional requirements; it is about requirements interaction analysis.
comment	4.15 This section is oriented towards CMM and away from open source project like Linux, that have very different processes.
CCSE response	We do not agree. The section is intended to cover a spectrum of life-cycle processes. Specifically the following are listed as examples: agile, heavyweight, waterfall, spiral, V-Model, etc.
comment	4.15 There should be a topic like PRO.con.8 which covers the interaction with users, such as how support groups feed user ideas into the development process, to deal with bugs and new features. It could also cover the release processes, which are very different than development processes.
CCSE response	A judgment has been made about what is appropriate and manageable for undergraduates. Unfortunately, not all topics desired by CCSE participants could be accommodated. Some of topics will have to be left for advanced training.
comment	4.15 Make PRO.imp.7, optional, because it is only one of many comparable processes. Perhaps it could discuss the diverse meta-processes instead. Perhaps it could also cover open source and other kinds of processes.
CCSE response	We have made PRO.imp.7 more general and have listed ISO/IEEE Standard as an example.
comment	4.17 Add a new topic MGT.con.6 on leadership.
CCSE response	Foundation leadership support is provide under "PRF.psy Group dynamics / psychology" and under "MGT.per Project personnel and organization". However, we believe an advanced topic on leadership would be appropriate after employment and when

	career goals are better established.
comment	5.1 The 2nd failure should be to fail to recognize the "software engineering nature" of software engineering. The current phrasing biases SE toward something that it is not.
CCSE response	We have changed "engineering-nature" to "engineering-oriented aspects".
comment	5.2 Guideline 4 The concept of an "engineering mindset" is very disturbing to me. I do not believe it exists as describe in here, or else computer science majors already have it. They are already able do understand measurement, modeling, human factors, scale, reuse, etc. And, while software engineering and traditional engineering share many common facets, they also have many major differences, which are not reflected here.
CCSE response	We fully agree that many computer science majors already have an engineering mindset. This must be true since much of the development of software engineering and large, complex, successful systems was and is done by those who have been educated in CS programs. However, not all CS graduates have the required background; and some CS courses or programs have tended to downplay engineering-oriented aspects. There are certainly cases where graduates of CS programs lack critical knowledge required to develop large, reliable systems. The fact that software engineering incorporates knowledge from diverse areas, is discussed elsewhere in the CCSE document. However, we believe that SE can rightly take its place among the other engineering fields – the fact that, for example, most other branches of engineering base designs on continuous, as opposed to discrete math, does not change the engineering nature of the design process used, unless you explicitly define engineering to require the use of continuous math – and we don't believe it is reasonable to do that. Similar arguments can be made for other ways in which SE differs from other engineering branches.
comment	5.3 Guideline 7 Software engineering is not necessarily a branch of traditional engineering. It should be taught so as to make software engineering into the best profession it can be, independent of the relationship to traditional engineering. While similarities with other branches of engineering should be recognized, the differences should also be recognized. The primary responsibility should be to the profession of software engineering, not to the profession of traditional engineering. Software engineers are real software engineers, and must develop a sense of the software engineering ethos. Note that the final sentence of this section argues that software engineering could also be like other professions, or

	independent. This guideline is very dangerous.
CCSE response	This point does not use the phrase traditional engineering. Nor do we believe there is such a profession as “traditional engineering”. We do realize that some non-software engineers do not appreciate the nature of software engineering. But that is a political issue; our job, as the reviewer says, is to help develop the profession of software engineering. Considering oneself an engineer (regardless of what other engineers may think) because one applies engineering principles is part of that. This point says, “recognize the similarities”, not “ignore the differences”.
comment	5.3 Guideline 8 Add cooperation skills, to the communication skills.
CCSE response	We agree. Change made.
comment	5.3 Guideline 11 This guideline states that everything students learn should be valid for “10 or 20” years which directly contradicts the notion that the field is evolving rapidly. Since most software engineers will not practice for more than 10 years, this means that continuing education is unnecessary. Perhaps we need to decide whether software engineering is evolving rapidly or not.
CCSE response	We have added the phrase “as much as possible”. We still believe, however, that a field can evolve rapidly, yet foundational knowledge and experience from older days can still be valid. For example, although one may have learned to program in COBOL many years ago, and does not use COBOL any more, having that knowledge still makes one feel he/she knows something about programming and languages that could be useful (it if is only, “don’t develop a language like that”).
comment	Guideline 12 I am not aware of any colleagues who ever had a serious problem learning new tools. Usually the problem is holding practitioners back, because the company needs compatibility between all projects for support purposes. I have never seen this bad habit in practice. This reinforces the mistaken notion that practitioners are incompetent and need brainwashing.
CCSE response	This point was added many iterations ago in response to contributors who felt that sometimes students are not exposed to tools, or only to arcane research ones (other than compilers and editors). We see your point, however, we don’t think we are trying to brainstorm practitioners; rather we are just trying to ensure that students use tools, for the reasons given.
comment	Guideline 18

	This is the classic “synergy” argument that you can magically get a lot of stuff for free. This argument helped Time Warner to buy AOL and Disney to buy ABC, which in retrospect were horrible decisions. I am very skeptical of this kind of argument.
CCSE response	Point taken. However, not all examples of synergy are horrible. Other reviewers and participants have agreed that in the context of education, this point is quite valid.
comment	5.5 Concluding Comment Change last sentence to read “with high-quality software engineering systems” to emphasize the real topic: software engineering.
CCSE response	The phrase “software engineering systems” is not a normal English phrase, whereas “software systems” is. We do not believe the change is justified.
comment	6.2 The arguments for the SE approach are biased toward engineering without proof that it is any better than the CS approach. The text associates “good habits” with SE and “bad habits” with CS. I am unaware that any CS educator ever attempted to teach anything but good habits. Certainly all of my professors tried their best. The text implies that Knuth and Dijkstra deliberately taught their students bad habits, which is bogus. Without any supporting studies of resulting behavior, the comparisons in these 2 paragraphs should be removed. **** Note that many engineering departments are now teaching project classes in the first 2 semesters as a way to motivate all of the theory that will come later. They noticed that dropping students directly into theory classes encourages them to switch to seek another major. Thus a better reason to teach the SE approach first might be “wholistic motivation”.
CCSE response	You are right to point out that it would be good if everything in the computing curricula documents were subject to empirical studies. Unfortunately such is not possible; therefore a lot of what you find in these documents is the consensus of many experts and reviewers such as yourself. We still agree with the spirit of the distinction between CS and SE courses, merely because many people have reported that plenty of CS students come out of CS1 and CS2 with a good ability to program in the small, but a tendency to not comprehend notions of scale, etc. Despite this, we have made some changes to tone down the wording. Note: We agree, that both, Knuth and Dijkstra were strong proponents of discipline (i.e., an engineering approach) in programming.

comment	6.2.1 There is no “engineering perspective” and both CS and SE can give programming assignments.
CCSE response	Sorry, we disagree – no change.
comment	6.3.2 There is too much formal methods and not enough practical stuff like refactoring.
CCSE response	It has been hard to achieve a balance. There remain those who feel there are not enough of lots of topics (including not enough formal methods). This is the typical “10 pounds into a 5 pound sack” problem. We all agree that we need to cover the fundamentals, but there is disagreement about what make up the fundamentals.
comment	6.4.1 Rename this “Software Engineering Economics” to reflect the software engineering nature of software engineering.
CCSE response	Many universities already offer a course like this. We felt it best to reuse where possible. Students will get lots of software engineering economics-type material in project management courses.
comment	6.4.2 Avoid engineering bias, and say “appreciation for software engineering in general.”
CCSE response	Here we are talking about taking courses outside software engineering.

reviewer	30000812753 Ricardo Colomo Palacios (Universidad Carlos III)
comment	Negative Comments Using of color-code in course coding scheme. I suggest using both color and plots instead.
CCSE response	Coloring schemes have been replaced with differences in shading, font, and borders.
comment	4.10 Include some references about dealing with multicultural environments
CCSE response	There is such a reference in Software Design. We agree that this is a missing element in Professional Practice and have added it as “PRF.psy.6 Dealing with multicultural environments”.
comment	4.11 Insert PRF.psy.5 as Related Topic for MAA.er.2
CCSE response	Recommendation incorporated.
comment	4.11 Delete MAA.er.2 as a Topic reference of itself
CCSE response	Correction made.
comment	4.17 Insert PRF.com.1, PRF.com.2, PRF.com.3 as a related topic for MGT.per.2
CCSE response	Good idea. Recommendations incorporated.
comment	4.17 Insert PRF.psy.1 as a related topic for MGT.per.1

CCSE response	Recommendation incorporated.
comment	4 Insert in SEEK a KU named MAA.mgt for Requirements management. In some course descriptions (SE324,SE322) there is a link to MAA.mgt and such KU is not present in the SEEK Essential
CCSE response	It should have been MAA.rfd.6 instead of MAA.mgt. Corrections made

reviewer	3000093713 Robert L. Glass (Computing Trends)
comment	Omissions There is nothing on these topics: teaching reading of artifacts before teaching about writing them (especially code) requirements-driven testing (this should be the minimal level of testing for all software products) structure-driven testing (this is obliquely covered by the discussion of test coverage, but it needs to be addressed more directly)
CCSE response	“Reading before writing” is a pedagogical approach that could clearly be adopted in CS 101. For example, such an approach is discussed in SE 102 under “Additional teaching considerations”. Also, Curriculum Guideline 4 encourages such an approach by the emphasis on studying existing software artifacts. Requirements-driven testing is covered under VAV.tst.8. VAV.tst.3 has been re-titled to “Coverage analysis and Structure Based Testing”

reviewer	30001015792 Kai Qian (Southern Polytechnic State U)
comment	Omissions The V & V case tools and evolution case tools such as tools for configuration management should be addressed. Software security should be emphasized in All phases.
CCSE response	Although tools are not explicitly addressed in the SEEK, Curriculum Guideline 12 in Chapter 5 asserts “Performing software engineering efficiently and effectively requires choosing and using the most appropriate computer hardware, software tools, technologies, and processes (again, collectively referred to as tools)”. This section describes issues and guidance for selecting appropriate tools. Section 4.18 on “Systems and Application Specialties” has several units on security and there are other references, implicit and e3xplicit, throughout the document (e.g., MAA.tm.2, MAA.af.3¸ DES.con.6, VAV.tst.9, etc.)

reviewer	30001127550 Maurizio Fenati (AICA)
comment	Omissions - Security basics as preliminary not only as domain specific (see SAS.net.3, SAS.fin.3, etc...).
CCSE response	Addressing security issues appear throughout virtually all of the KAs.
comment	FND.mf FND.mf (Mathematcial fundations) It has few hourse (only 56) considering its importance; I will add Logic arguments. Indeed Algorithms, Data Structures/Representation (static & dynamic) and Complexity should be reviewed/assessed (unless CMP.cf is pre-requiste to FND.mf) in this context in order to introduce Combinatorial and Optimizations problems (primal & dual simplex, Lagrange curve utilization, etc...).
CCSE response	These hours are particular to Mathematical Foundations, which are part of the SE core. There will certainly be more hours devoted to mathematics as part of the degree.
comment	PRF.pr PRF.pr (Professionalism) I will mention Open Source definition and GNU General Public License.
CCSE response	These topics could be raised as part of PRF.pr.3 and PRF.pr.6.

reviewer	30001214638 Jurgen Borstler (Umea University)
comment	Appendix A: SE400 Required team size should be at least 4. Teams of two can be easily confused with the currently so popular pair programming approach. Two people are usually not sufficient to experience team dynamics. In the literature 4-7 is often mentioned as a common team size.
CCSE response	Since there are enough people who feel that a capstone project can be done with fewer than four, we are leaving this unchanged. In fact, we received a comment by another reviewer who insisted that capstone projects be done individually.

reviewer	30001328441 Ana Moreno (Universidad Politecnica de Madrid)
comment	International adaptation issues Main differences among the different countries, mainily in English speaking countries and european countries is the accreditation police, in most european countries (with the exception of UK) accreditation is done before a program is running and it is done usually by the State. Therefore in Section 8.2. it might be worthwhile to mention that the accreditation issues mentioned in it refer to those countries where accreditation is done after a program is running. But there are many other countries where such accreditation is done, by the State according to an official syllabus, before a program starts

	running. However, CCSE might also be used in those cases as an inspiration for any official syllabus.
CCSE response	Good point. The recommendation was incorporated into the discussion in section 8.2 of assessment and accreditation.
comment	4.10 Increase the hours assigned to Professional Practice. Teaching about teamwork, negotiation and stakeholders interaction techniques are important for developers and of practical utility in their professional practice. I guess that if we compare such knowledge with for example, professionalism, although the last ones are also relevant, possibly the first ones have more practical utility and have 1/4 from the time assigned to professionalism.
CCSE response	You will find some of the issues you raised in other KAs such as MAA and MGT, which increases the amount of time devoted to such activities.
comment	6.3 Explicitly mention the reasons that provoke the two core SE course sequences. In the document it is mentioned might be different course alternatives for a SE undergraduate program but the final knowledges should be similar. In the second approach I did not identified Requirements course related. Is there any reason for that?
CCSE response	We have modified this section to address these concerns.

reviewer	30001521698 Rick Duley (Murdoch University)
comment	NT181 etc. If NT272 and NT291 were amalgamated to one full course and NT181 expanded to another full course I think the balance would be better. Otherwise, great job folks! As we say in Oz, "Gudonyer!"
CCSE response	An institution could certainly do this. However, many institutions have existing courses, especially for NT 272 and NT291, which we think do a good job.

reviewer	3000166654 Bill Hefley (Carnegie Mellon University)
comment	2.2 [5] pp 12 Add "documentation, " after "implementation, "
CCSE response	The addition was incorporated.
comment	2.2 p 13 Add "[8] Learn to practice software engineering with an appreciation of ethical, privacy, and security issues."
CCSE response	Wording was added to outcome [1], which emphasizes that it is important that SEs appreciate and understand issues related to

	ethics and professional conduct, societal needs, etc. Security was addressed under [4].
comment	4.9 Add FND.ec5 Risk management Inadequate preparation in foundations to specifically deal with probabilistic risk analysis. Needed as foundation for software management topic on risk management.
CCSE response	This topic appears in both MAA and MGT (MGT.pp.6 Risk management). It would also be reasonable to teach as part of FND.ec.1.
comment	Chapter 5 - Guideline 4 Add "ethics" as a bullet in this list Ethical considerations is not a topic that should be addressed once as an isolated management or professionalism topic, but must be repeatedly addressed in the context of numerous systems and software engineering activities. Consistent with Guideline 15.
CCSE response	We certainly agree with the point, but believe that guideline 15 already does what is needed. Guideline 4 (just before the bullets) points out that the bulleted list is “in addition to ethics … which will be highlighted specifically in other guidelines”
comment	Chapter 5 - Guideline 8 Add "Behaving ethically and professionally. Students should learn to behave ethically and to understand the ethical, privacy, and security implications of their work."
CCSE response	We agree, and have added a bullet with the recommended wording.

reviewer	James Moore (Mitre Corporation)
comment	4.6 Selection of Knowledge Areas "Although the SWEBOK did serve as a starting point for determining knowledge areas, both the CCSE Steering Committee and the SEEK area volunteers felt strongly about emphasizing the academic discipline of software engineering. During the SEEK development process, the area chosen to represent the theoretical and scientific foundations of developing software products subsequently grew to the size of one half of the core. This prompted the Steering Committee to reevaluate whether the original goals of emphasizing the discipline were indeed being met. The resulting set of knowledge areas are believed to stress the fundamental principles, knowledge, and practices that underlie the software engineering discipline." Terms like "although" and "but" provide a sense of repudiation, although I doubt that any is intended. I would like to suggest a minor rephrasing of this passage as follows:

	<The SWEBOK Guide provided our starting point for determining knowledge areas. Because both the CCSE Steering Committee and the SEEK area volunteers felt strongly about emphasizing the academic discipline of software engineering, the area chosen to represent the theoretical and scientific foundations of developing software products eventually grew to one half the size of the core. This prompted the Steering Committee to reevaluate whether the original goals of emphasizing the discipline were indeed being met. The resulting set of knowledge areas were rebalanced to support these goals. The result is believed to stress the fundamental principles, knowledge, and practices that underlie the software engineering discipline in a form suitable for undergraduate education.>
CCSE response	We think the recommendation is excellent and have incorporated it in the CCSE volume.