THE STATE OF SOFTWARE ENGINEERING MATURITY AND LICENSURE

A Thesis
Presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

by
Travis Lawrence Dean
March 2011
COMMITTEE MEMBERSHIP

TITLE: The State of Software Engineering Maturity and Licensure

AUTHOR: Travis Lawrence Dean

DATE SUBMITTED: March 2011

COMMITTEE CHAIR: Dr. Clark Savage Turner
Software Engineering Coordinator
Computer Science Professor
Computer Science Department
California Polytechnic State University

COMMITTEE MEMBER: Dr. Gene Fisher
Computer Science Professor
Computer Science Department
California Polytechnic State University

COMMITTEE MEMBER: Dr. David Janzen
Computer Science Associate Professor
Computer Science Department
California Polytechnic State University
Abstract

The State of Software Engineering Maturity and Licensure

by

Travis Lawrence Dean

IEEE-CS is pursuing licensing for software engineers, but ACM believes that software engineering is too immature, and regulating the profession would be premature. In 1996, Norman Gibbs and Gary Ford from Carnegie Mellon University performed a qualitative survey of the maturity of the software engineering profession. I apply this model to the present state of the art in software engineering and analyze the results for 2010. I analyze the maturity of software engineering to determine that the profession is not yet ready for licensure. This is not because the infrastructure of software engineering is too immature, but because we have failed to establish an appropriate body of knowledge for software engineers. I also show that once an appropriate body of knowledge is established, licensure will be an appropriate next step and will open the way for the profession to fully mature.

Keywords: Software engineering, maturity model, professional, profession, licensing.
Acknowledgements

There are many people who helped to make this thesis possible.

Jennifer Elizabeth Archer for reading over my paper for clarity and grammar usage and also for supporting me through this project.

David Lorge Parnas for his email correspondence which ultimately led me to taking the direction my thesis went.

Clark Savage Turner for guiding me through writing my thesis, suggesting works and professionals to look at to perform my analysis, and for supporting me when I ran into difficulties with my research.
# Table of Contents

List of Tables ................................................................................................................................ ix
List of Figures .................................................................................................................................. xi

1. **Introduction** ........................................................................................................................... 1
   1.1. Scope ................................................................................................................................ 1
   1.2. Contribution ..................................................................................................................... 2
   1.3. Organization of Thesis ..................................................................................................... 2

2. **Background** ............................................................................................................................ 3
   2.1. Current Professional Licensing .................................................................................... 3
       2.1.1. History of Licensing ................................................................................................. 3
       2.1.2. Purpose of Licensing ................................................................................................. 6
       2.1.3. Becoming Licensed ................................................................................................... 6
       2.1.4. Who Requires Being Licensed .................................................................................. 7
       2.1.5. Responsibility and Malpractice ............................................................................... 10
   2.2. A Software Engineering Timeline .................................................................................. 11
       2.2.1. Software Engineering is not Computer Science ..................................................... 11
       2.2.2. ACM and IEEE Form a Joint Committee ............................................................... 15
       2.2.3. Texas Implements Licensing for Software Engineers ............................................ 15
       2.2.4. Current Licensing Progress for Software Engineers ............................................... 16

3. **Maturity Model of a Profession** .......................................................................................... 19
   3.1. Qualifying the Evolutionary Stages ............................................................................... 22
   3.2. Initial Professional Education ........................................................................................ 22
   3.3. Accreditation of Professional Education Programs ....................................................... 24
   3.4. Skills Development ........................................................................................................ 24
   3.5. Certification .................................................................................................................... 25
   3.6. Licensing ........................................................................................................................ 26
   3.7. Professional Development.............................................................................................. 28
   3.8. Code of Ethics ................................................................................................................ 29
   3.9. Professional Society ....................................................................................................... 29
   3.10. Body of Knowledge ...................................................................................................... 30
   3.11. Summary of the Maturity Model of a Profession ........................................................... 32
4. The Maturity Model Applied to Software Engineering ................................................. 37

4.1. Initial Professional Education .................................................................................. 37
   4.1.1. 1996 – Initial Professional Education ............................................................... 38
   4.1.2. 2000 – Initial Professional Education ............................................................... 38
   4.1.3. 2004 – Initial Professional Education ............................................................... 39
   4.1.4. 2010 – Initial Professional Education ............................................................... 40
   4.1.5. Summary – Initial Professional Education ....................................................... 41

4.2. Accreditation .......................................................................................................... 42
   4.2.1. 1996 – Accreditation ......................................................................................... 42
   4.2.2. 2000 – Accreditation ......................................................................................... 43
   4.2.3. 2004 – Accreditation ......................................................................................... 43
   4.2.4. 2010 – Accreditation ......................................................................................... 44
   4.2.5. Summary – Accreditation ................................................................................. 44

4.3. Skills Development ............................................................................................... 45
   4.3.1. 1996 – Skills Development ................................................................................. 45
   4.3.2. 2000 – Skills Development ................................................................................. 46
   4.3.3. 2004 – Skills Development ................................................................................. 46
   4.3.4. 2010 – Skills Development ................................................................................. 47
   4.3.5. Summary – Skills Development ....................................................................... 48

4.4. Certification ............................................................................................................. 48
   4.4.1. 1996 – Certification ............................................................................................ 48
   4.4.2. 2000 – Certification ............................................................................................ 49
   4.4.3. 2004 – Certification ............................................................................................ 49
   4.4.4. 2010 – Certification ............................................................................................ 51
   4.4.5. Summary – Certification .................................................................................. 51

4.5. Licensing .................................................................................................................. 51
   4.5.1. 1996 – Licensing ................................................................................................ 52
   4.5.2. 2000 – Licensing ................................................................................................ 53
   4.5.3. 2004 – Licensing ................................................................................................ 54
   4.5.4. 2010 – Licensing ................................................................................................ 54
   4.5.5. Summary – Licensing ...................................................................................... 55

4.6. Professional Development ....................................................................................... 55
   4.6.1. 1996 – Professional Development ..................................................................... 56
   4.6.2. 2000 – Professional Development ..................................................................... 57
4.6.3. 2004 – Professional Development ................................................................. 57
4.6.4. 2010 – Professional Development ................................................................. 58
4.6.5. Summary – Professional Development .......................................................... 58
4.7. Code of Ethics ..................................................................................................... 59
  4.7.1. 1996 – Code of Ethics .................................................................................. 59
  4.7.2. 2000 – Code of Ethics .................................................................................. 60
  4.7.3. 2004 – Code of Ethics .................................................................................. 61
  4.7.4. 2010 – Code of Ethics .................................................................................. 61
  4.7.5. Summary – Code of Ethics .......................................................................... 62
4.8. Professional Society ............................................................................................ 62
  4.8.1. 1996 – Professional Society ......................................................................... 63
  4.8.2. 2000 – Professional Society ......................................................................... 63
  4.8.3. 2004 – Professional Society ......................................................................... 64
  4.8.4. 2010 – Professional Society ......................................................................... 64
  4.8.5. Summary – Professional Society ................................................................. 65
4.9. Body of Knowledge ............................................................................................. 65
  4.9.1. 1996 – Body of Knowledge ......................................................................... 65
  4.9.2. 2000 – Body of Knowledge ......................................................................... 66
  4.9.3. 2004 – Body of Knowledge ......................................................................... 67
  4.9.4. 2010 – Body of Knowledge ......................................................................... 68
  4.9.5. Summary – Body of Knowledge .................................................................. 69
4.10. Summary of Software Engineering Maturity ..................................................... 69
4.11. Threats to Validity ............................................................................................. 71
5. Next Steps ............................................................................................................. 73
  5.1. Principle and Practices of Engineering Exam .................................................... 73
  5.2. Fundamentals of Engineering Exam ............................................................... 74
  5.3. Body of Knowledge ......................................................................................... 75
6. Conclusion ............................................................................................................. 78
References ............................................................................................................... 79
List of Tables

Table 2-1 – The percentage of engineers seeking licenses according to a 1996 study....... 7
Table 3-1 - The components of a profession. ................................................................. 20
Table 3-2 - Evolutionary stages to characterize the infrastructure-level components..... 22
Table 3-3 - Entry level professions and degree requirements................................. 23
Table 3-4 – The percentage of engineers seeking licenses according to a 1996 study.... 27
Table 3-5 – Projected evolution of software engineering as a profession............... 33
Table 3-6 - Rubric for measuring maturity of the body of knowledge......................... 34
Table 4-1 - Rubric for initial professional education.................................................. 37
Table 4-2 - Summary of maturity progress for initial professional education.............. 42
Table 4-3 - Rubric for accreditation........................................................................... 42
Table 4-4 - Summary of maturity progress for accreditation. .................................... 44
Table 4-5 - Rubric for skills development. ................................................................. 45
Table 4-6 - Summary of maturity progress for skills development............................ 48
Table 4-7 - Rubric for certification............................................................................. 48
Table 4-8 - Summary of maturity progress for certification....................................... 51
Table 4-9 - Rubric for licensing. ................................................................................. 52
Table 4-10 - Summary of maturity progress for licensing......................................... 55
Table 4-11 - Rubric for professional development..................................................... 56
Table 4-12 - Summary of maturity progress for professional development............... 59
Table 4-13 - Rubric for code of ethics. ................................................................. 59
Table 4-14 - Summary of maturity progress for code of ethics................................. 62
Table 4-15 - Rubric for professional society. ............................................................. 62
Table 4-16 - Summary of maturity progress for professional society. ......................... 65
Table 4-17 - Rubric for body of knowledge. ................................................................. 65
Table 4-18 - Summary of the maturity progress for a body of knowledge .................. 69
Table 4-19 - Maturity level of each content area over time ....................................... 70
Table 4-20 - Meaning of the maturity rating values. .................................................... 70
List of Figures

Figure 1 - Flowchart depicting when a professional engineer is required ......................... 9

Figure 2 - Interactions among components of a profession ............................................. 21

Figure 3 - Maturity of the components over time ............................................................ 70

Figure 4 - Number of ABET accredited software engineering programs by year .......... 74
1. Introduction

Software engineering is a young profession which has developed significantly over the years. Today, software has found its way into medical devices, aircraft, missiles, and other real-world devices that can cause significant harm to the public if handled improperly (1)(2)(3). However, software engineering is not considered a mature profession like other disciplines in the fields of engineering, medicine, and law (4; 5). IEEE-CS and ACM have made strides towards the professionalization of software engineering (6). IEEE supports licensure of the profession. However, ACM believes software engineering is too immature for licensing.

The question of “How mature is software engineering?” is not new. In fact, Gibbs and Ford from Carnegie Mellon University developed a model to characterize the maturity of professions in 1996 (5).

1.1. Scope

I will review how other engineering professionals are licensed, how software engineering has developed and matured over time, and I will analyze and update the application of the Gibbs model of maturity to the current state of the art in software engineering. I focus on software engineering as a profession inside the United States specifically.

In this thesis, I use the definition of maturity as “The state or quality of being fully developed” (7).
1.2. Contribution

There is a lot of controversy over whether or not software engineers should be licensed and how they should be licensed. A lot of the debate comes from a misconception of how licensing would apply to software engineering and what would be the effects of licensing. I explain the implications of licensing to software engineering to clarify the issue. This thesis contributes the tradeoffs of licensing for software engineers and whether now is an appropriate time to institute licensing.

Much work and progress has been made with regard to various aspects of software engineering licensure and maturity. I have pulled together the content from papers and investigations and compiled the results of my research in this thesis. I use this to provide an updated qualitative measure of the current status of software engineering’s maturity based on the rubric provided by Gibbs and Ford from Carnegie Mellon University. I also contribute the addition of a body of knowledge to this rubric.

1.3. Organization of Thesis

This thesis is outlined into 6 chapters as follows:

- Chapter 1 is the introduction.
- Chapter 2 covers the history of licensure for engineering and surveyors and a brief timeline of software engineering.
- Chapter 3 presents a professional maturity model developed to qualitatively measure the maturity of a profession.
- Chapter 4 applies the maturity model to software engineering.
- Chapter 5 presents the next steps in the licensing process for software engineers.
- Chapter 6 summarizes the findings of this report and conclusions of my survey.
2. Background

To make reasonable sense of this thesis we need a clear understanding of what professional licensing is, how it works, what specifically software engineering is, and update our current understanding of the state of software engineering. In this chapter, I provide background on licensing and how it potentially applies to software engineering. Then, I provide a timeline of software engineering events relevant to its maturity.

2.1. Current Professional Licensing

In this section, I briefly describe the history of licensing. Then, I go into some detail about the purpose and process of becoming licensed, which engineers get licensed, and the responsibility that comes with becoming licensed.

2.1.1. History of Licensing

Before licensing laws, anybody could claim to be an engineer and practice as one regardless of their experience level and qualifications. In the late 1800s and early 1900s, before licensing laws were in place, there was no distinction between professional engineers and untrained, unqualified individuals (8). The necessity for changes in these practices became apparent when engineers became involved in building public structures such as the Transcontinental Railroad and the Brooklyn Bridge where lives of numerous members of the public depended on the professional care put into such structures.

In 1907, Wyoming passed the first law requiring registration for people who would represent themselves as engineers or surveyors to the public. Wyoming also created a state board of examiners for the profession in the same year (8)(9). Three more states passed an engineering registration law the following year, with six more states
passing such a law in 1915. As these additional states started passing their own
ing engineering laws, it became apparent that the language of these laws varied greatly from
state to state. In 1920, the Iowa state board called for a meeting of the ten existing boards
to “create an organized and systemized method of procedure to be followed in interstate
registration” (8). This council coordinated reciprocal relations between the member
states, meaning that licensed engineers possessing a reciprocal card issued by a licensing
board would be recognized by all member states.

The reciprocal cards required that all states pass licensing laws and join this
committee. By 1947, Montana became the last state to enact licensing laws, and in 1950,
the entire country had some form of licensing laws (8)(9). Despite this, licensure
examinations differed greatly from state to state. Some states required only an oral
examination, whereas others required only written exams. Some exams could be
answered with a solid understanding of logic, but did not require any engineering
knowledge, and others required the memorization of complex equations (8). Therefore,
the exams had not established consistent standards across the country for competence.

State boards responded by developing questions according to a rational method
that would require test takers to demonstrate their ability to think like engineers (8). The
boards proposed a two part exam. The first part would last two and a half days and cover
fundamentals such as math, applied science, electrical and machine design, and
engineering economics, law, and practice. The second part was to last half a day and
cover one of five specialty engineering fields: chemical, civil, electrical,
mechanical/industrial and mining/metallurgical (8). State boards eventually approved
this, though with some modifications. Many boards instituted a new category of
registration for engineers without experience who were interested in becoming a licensed engineer. This was the Engineer in Training exam, the first part of which was administered to college seniors and those interested in initiating the process (8). This later came to be known as the Fundamentals of Engineering (FE) exam (8)(9).

In May 1965, the first FE exam was administered in 30 states. This exam contained 30 essay questions covering 10 subjects. The following year, a national uniform Principles and Practices of Engineering (PE) exam was made available and administered (8)(9). The essay questions on the FE and PE exams have since been replaced with multiple-choice questions developed by committees, graded by computers, and psychometrically tested for fairness and relevance. The exam items are written by volunteer licensed engineers from academia, consulting, and industry (8). It is the professionals that govern their own standards. They perform extensive surveys of professionals across a range of industries, which are used to determine exam specifications (8).

In 1984, all member boards began administering uniform national engineering examinations. In 1996, the afternoon portion of the FE exam started optionally including one of six discipline-specific modules to test upper-division knowledge (9). These modules are to test upper-division knowledge and include one of chemical, civil, industrial, electrical, mechanical, or general for the discipline-specific module (10).

Licensing of engineers has undergone many modifications, improvements, and standardizations since it first started over a century ago (10). All the changes that have occurred have been to further assure that professionals are capable and competent to protect public health, safety, and welfare (10).
2.1.2. Purpose of Licensing

The ultimate goal of licensure is to protect public health, safety, and welfare (11). When licensing was first initiated, the intent was to distinguish a trained professional from an untrained, unqualified individual when lives were at stake (8). The requirements of licensing also set the standard for how competent a professional engineer must be and gives definition to how a trained professional would act (12). This lets us define the minimum competence required of an engineer to perform safe practices and provides established standards for determining when malpractice is performed by a registered professional.

2.1.3. Becoming Licensed

Licensure is a mandatory process administrated by the state government. The standards that the professionals are held to are defined by the profession itself. The laws reflect these standards set by the professionals. While licensing laws are state-specific, a voluntary organization of professionals called the National Council of Examiners for Engineering and Surveying has established a model law which is now adhered to by each U.S. state and territory. For an individual to become licensed as a professional engineer, he or she must meet the following requirements in order (10):

1. Graduate from an ABET accredited program
2. Pass a Fundamentals of Engineering (FE) exam
3. Practice as an engineer in training for four years
4. Pass a Principles and Practices of Engineering (PE) exam

For a new profession to become recognized and eligible for licensing its practitioners, NCEES has the following suggested requirements which an engineering profession must have established (10):

1. Code of Ethics
2. Esoteric Body of Knowledge
3. ABET Accreditation of undergraduate programs

2.1.4. Who Requires Being Licensed

An important aspect of licensure is distinguishing who should be licensed amongst engineers. One misconception about licensing is the concept that all engineers need to be licensed (4). Table 2-1 shows the percentage of engineers seeking an engineering license in 1996 (13). This shows that even the most frequently licensed profession, civil engineering, has a licensure rate of less than fifty percent. The two professions in which practitioners are commonly employed in larger organizations, electrical and chemical engineering have fewer than ten percent of engineers seeking licensure.

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Licensed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civil</td>
<td>44</td>
</tr>
<tr>
<td>Mechanical</td>
<td>23</td>
</tr>
<tr>
<td>Electrical</td>
<td>9</td>
</tr>
<tr>
<td>Chemical</td>
<td>8</td>
</tr>
<tr>
<td>All Engineers</td>
<td>18</td>
</tr>
</tbody>
</table>

According to the National Society of Professional Engineers (NSPE), the distinction between a registered professional engineer (PE) and an unlicensed engineer is that a PE must continually demonstrate their competency and maintain and improve their skills by fulfilling continuing education requirements dictated by the state in which they are licensed (14). NSPE also makes the following notes on what distinguishes PEs from other engineers¹ (14):

---

¹ I have italicized key words and phrases that are important to take note of in this list.
Only a licensed engineer may prepare, sign, seal, and submit engineering plans and drawings to a public authority for approval, or seal engineering work for public and private clients.

Professional engineers shoulder the responsibility not only for their work, but also for the lives affected by that work and must hold themselves to high ethical standards of practice.

Licensure for a consulting engineer or a private practitioner is not something that is merely desirable; it is a legal requirement for those who are in responsible charge of work, be they principals or employees.

Licensure for engineers in government has become increasingly significant. In many federal, state, and municipal agencies, certain governmental engineering positions, particularly those considered higher level and in responsible positions, must be filled by licensed professional engineers.

Many states require that individuals teaching engineering must also be licensed. Licensure also helps educators better prepare students for their future in engineering.

By applying these guidelines to software engineers, licensing would be required of those practicing as consulting engineers or private practitioners, and in certain high-level positions in government, in addition to those who sign and seal engineering work for public and private clients, and furthermore may someday be required for individuals teaching software engineering. This is critical for life critical public works where people must rely on the work of an engineer and can’t be responsible to analyze their safety as a common citizen (14). The professional engineer must be responsible because only the professional engineer has the knowledge and skills to understand the quality of their work (14).

In practice, most engineers don’t seek licensure because they don’t need to. An engineer can work for a large corporation without having to be licensed (5). Additionally, an engineer can work for under a licensed professional engineer. This will leave the licensed professional responsible for the work of the employee working for him (5).

To become more precise in defining who needs to be licensed in a practical sense, I will use Figure 1 (15). Many states define specific requirements of when a registered
professional engineer is required for primarily civil, mechanical, and electrical engineers (16). Figure 1 shows that the need of a licensed engineer of any of these three disciplines is typically for building construction, and even then, there are exceptions.

Figure 1 - Flowchart depicting when a professional engineer is required.

There is no equivalent chart of this nature for software engineers to define when a software engineer is required to be licensed (17; 18). I confirmed this with David Howell, who is the Director of Licensing on the Texas Board of Professional Engineers. The development of guidance papers for licensure issues are under development by a team formed in part by the Texas Board (17). Therefore, I can’t say distinctively at this time when a licensed software engineer is required.
2.1.5. Responsibility and Malpractice

As registered professionals, engineers hold a responsibility to protect the public (10). When an action performed by an engineer has potential to cause or has caused harm to the public, that engineer is at risk of a liability lawsuit. Some software engineers oppose licensing because of this liability (4). When engineers and surveyors were first being licensed, they opposed licensing for the same reason; they were afraid of being held liable for their actions (8).

When accused of malpractice, a court determines whether a professional was negligent in his actions. If so, then he is found legally liable and must pay for the damages caused by his negligence (19). In this section, I look at how negligence is determined and what are the potential consequences.

2.1.5.1. Determining Negligence

In legal terms, negligence is defined as behavior which a reasonably prudent person in the same or similar circumstances would not have undertaken (19). The behavior a reasonable person would have undertaken is defined in many ways. A professional society that represents the profession itself defines or maintains the tools used to define how a reasonable professional would act (20). These tools could include a body of knowledge, a code of ethics and professional practice, and are generally taught through initial professional education (20).

2.1.5.2. Consequences of Malpractice

The consequences for a software engineer found liable for malpractice are similar to those of many other professions. After being found liable in a court case, the engineer must compensate the victim financially for the malpractice. Other consequences are
determined by a governing society of that profession (12). Software engineering has an established code of ethics; however, there are currently no established consequences for a software engineer who has committed malpractice (19). Other professions may serve as a model in that consequences to malpractice include revocation of a license or additional fines (14). Since malpractice is covered under civil law, not criminal law, malpractice suits are only about money damages.

2.2. A Software Engineering Timeline

The term ‘software engineering’ came into popular use at a 1968 NATO conference on software engineering (21). Since then, software engineering has grown in verbal use and practice. However, it wasn’t until about 1993 when software engineering started to mature as a profession (22). This section covers history of the development of software engineering as a profession. The progress that has been made towards software engineering maturity is further expanded upon and qualified in chapter 4 to show that software engineering has matured to the point where licensing will be beneficial to its professionalization.

To fully understand where software engineering is in its maturity, we need to explicitly define what is software engineering. A critical aspect to this definition is making clear the difference between software engineering and computer science because many individuals disagree on what these differences are or if differences even exist (23).

2.2.1. Software Engineering is not Computer Science

The term “software engineering” has grown in use since the 1968 NATO conference. Many individuals, however, disagree on what software engineering is. Some believe it to be a subset or a specialization of computer science (24). Others believe it to
be a separate discipline altogether which applies the theory from computer science (23). Often, software engineering and computer science are even used interchangeably (23). In many schools, the difference between a software engineering program and a computer science program is small (24).

According to Steve McConnell in 2004, about 40% of software developers in industry held computer science degrees and almost none held software engineering degrees (25). This shows that only a small fraction of individuals who are developing software have formal education in engineering. Licensure would ensure that individuals in charge of safety-critical and mission-critical software projects are ones with an engineering background who have demonstrated minimal competence in software engineering (10).

2.2.1.1. Engineering vs. Science

I will start differentiating computer science from software engineering by examining their labels. One is labeled engineering, and the other is labeled a science, so we will start by distinguishing scientists from engineers. Scientists learn what is true, how to test hypotheses, and how to extend knowledge in their field. Engineers learn what is true, what is useful, and how to apply well-understood knowledge to solve practical problems (25). A science education prepares students to extend our knowledge in a field, whereas engineers are trained to develop products and techniques for a specific market or client (23). The difference is the application of knowledge in order to extend our knowledge versus the application of knowledge for the development of products.

The Texas Licensing Board came to define the “practice of engineering” as follows (26):
The “practice of engineering” means any service or creative work, the adequate performance of which requires engineering education in the application of special knowledge of the mathematical, physical, or engineering sciences to such service or work.

2.2.1.2. Defining Software Engineering and Computer Science

Once we have grasped the distinction between engineering and science, we can use this foundation to examine definitions of software engineering and computer science which correlate to the differences in the two disciplines. Because this thesis deals with determining the maturity of software engineering for licensure, I am defining software engineering and computer science as they should be defined according to ACM and IEEE-CS, not necessarily as they are treated in today’s world. The ways in which they are treated today contain too much variance and overlap between the two disciplines to easily distinguish them from each other (23).

Here I give a definition of software engineering which I adopt for use in this paper:

“Software Engineering is the application and/or study of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software that has an impact on the lives, property, economy, or security of people or the national defense; that is, the application of engineering to software” (27).

For contrast, here is a definition of what computer science is:

“Computer Science is the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society” (28).

Take note of how computer science is defined as the study of aspects relating to computers, and software engineering is the application of aspects relating to computers, much like in how we defined the differences in engineering and science in the previous section. In practice today, education programs don’t necessarily make such clear-cut
distinctions between computer science and software engineering. As a result, many practicing software engineers can come from either a computer science program or a software engineering program (25). Depending on the school implementing the program, either program can have either a science focus or an engineering focus, or a blend of both (24).

2.2.1.1. Similarity to Physics and Electrical Engineering

Dave Parnas has related the differences in software engineering and computer science to the differences in electrical engineering and physics (23). In the latter part of the 1800s, electrical engineering was considered a subfield of physics (29), much like many schools consider software engineering a subfield of computer science (23). It wasn’t until late in the 1800s when universities started offering a degree in electrical engineering. In 1882, the first electrical engineering program was offered (29).

The question arose for electrical engineering, “Why not just study Physics?” (23). The biggest reason is that they lead to two very different career paths (29). One is to produce graduates who will design products for others to use. The other path is to produce graduates who will study phenomena which are of interest to both groups and extend the knowledge of both areas (23).

Software engineering is different from computer science in the same sense, which correlates to the same difference between scientists and engineers. Scientists learn science plus scientific methods needed to extend it. Engineers learn science plus the methods needed to apply it (23). However, not all education programs grasp this distinctive difference between the two disciplines. Parnas proposes that software engineering courses should even be taught with a different emphasis than the equivalent
course for a computer scientist (30). Note that ACM and IEEE-CS have made joint efforts in this direction for developing a distinctive initial education for software engineering (31).

### 2.2.2. ACM and IEEE Form a Joint Committee

In 1993, the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) began a joint pursuit towards professionalizing software engineering (4). Their joint committee came to be known as the Software Engineering Coordinating Committee (SWECC) (32). SWECC ultimately developed the Software Engineering Body of Knowledge (SWEBOK), the Software Engineering Code of Ethics and Professional Practice, guidelines for developing initial professional education in software engineering, and more (33; 34; 31).

In 1998, Barbara Simons, the president of ACM, formed an ACM Advisory Panel on Professional Licensing of Software Engineers to determine whether ACM should continue with the SWECC efforts (35). While the opinions of the committee members were split almost evenly, the majority of the committee members concluded that licensing was premature for software engineers and it would not be effective for addressing the problems of software quality and reliability (4). Based on these findings, ACM backed out of SWECC in 1999 and openly opposed any form of licensing of software engineers (8).

### 2.2.3. Texas Implements Licensing for Software Engineers

In 1997, the Texas Licensing Board formed a panel to discuss licensing for software engineers working on safety-critical applications (36). I could not find direct evidence of any single accident triggering this motion, but a very likely incident is the
Therac-25 accident where software errors contributed to the failure of the system (Therac-25) (37). Some of the board’s concerns included a “22 year old college graduate claiming to be an experienced software engineer” and “systems that do not accomplish what they claimed they could do” (38).

At first, the Texas Board considered software engineering a sub-discipline of electrical engineering, much like they considered computer engineering to be a sub-discipline of electrical engineering. Fortunately, the board formed a software engineering advisory committee to help them better understand the field of software engineering.

This advisory committee led the board to instituting licensing for software engineers as a separate profession from electrical engineering and other engineering disciplines (36). However, software engineering doesn’t have its own PE licensing exam (39). Additionally, software engineering hadn’t fully established a code of ethics, a body of knowledge, or any of the other standard requirements by which a profession validates licensure (14). Due to this lack of maturity in the profession, software engineers could only be licensed in Texas through Texas’s exam-waiver clause. To obtain a PE license before the exam becomes available, an applicant must have one of the following (40):

- 16 years of engineering experience.
- 12 years of engineering experience and a bachelor’s degree from an accredited university program.
- 6 years of experience and a Ph.D. in engineering or a related subject from a university whose undergraduate program is accredited.

To date, Texas has licensed 57 software engineers by these means (41).

2.2.4. Current Licensing Progress for Software Engineers

Seventeen years have passed since ACM and IEEE started the pursuit of software engineering professionalization and it has been thirteen years since Texas established a software engineering as a separately licensed discipline (40). To date, Texas is still the
only state in the U.S. to license software engineers. Since there still is not a P.E. exam for software engineering, professionals are still licensed in Texas through the exam-waiver clause (40). Software engineers are also licensed in Australia and some provinces and territories of Canada (9).

Following are the NSPE’s suggested standard requirements for a discipline to qualify for licensure (14):

1. ABET recognition and accreditation
2. An esoteric body of knowledge
3. Code of ethics
4. Principles and practices of engineering exam

In 1998, ABET recognized and accredited software engineering as an undergraduate engineering program (42). The first accredited software engineering undergraduate program became accredited in 2002 (43). IEEE-CS released the first version of a software engineering body of knowledge in 1999 and a revised edition in 2004 (44). The ACM and IEEE-CS Joint Task Force established a code of ethics and professional practice for software engineers in 1999 (34).

The only remaining item to be implemented is a software engineering principles and practices of engineering exam. IEEE reported in September 2009 that their organization was approved by NCEES to develop an exam requested by state licensure boards for prospective use in licensing software engineers (12). Once this PE licensing exam is completed, software engineering will have all of the requisite materials to be eligible for having licensed practitioners.

While IEEE-CS continues their pursuits towards licensing, ACM’s maintains that software engineering is too immature to be licensed. In the next chapter, I look at a maturity model which I will use to characterize software engineering’s maturity. Using
this analysis of software engineering’s maturity, I will answer whether or not software engineering is mature enough for licensing.
3. Maturity Model of a Profession

Gary Ford and Norman E. Gibbs from Carnegie Mellon University performed an analysis of mature professions and examined what each profession had in common. From this analysis, they extracted eight infrastructure-level and three practitioner-level components common to most professions as shown in Table 3-1.

Ford and Gibbs were unable to use the practitioner level components of a profession in any meaningful way to measure the maturity of that profession. For professionals, they could not come up with any useful assessment that would describe the maturity of a profession (5). Ford and Gibbs could not figure out any way to characterize knowledge because a body of knowledge for any given profession continues to evolve over time (5; 4). There is no definable point at which it changes from immature to mature. Professional practice depends on a body of knowledge, so it evolves as the body of knowledge changes. This makes professional practice as impossible to characterize as a body of knowledge (5).

Therefore, the infrastructure level components were the only components used to formulate a maturity model. Using this model, Ford and Gibbs were able to analyze the maturity of software engineering. This study was performed for the Software Engineering Institute to characterize and model the evolution and maturation of professions and to predict how the software engineering profession might develop (5).

This study provided what Ford and Gibbs believed to be an appropriate vision for the future of software engineering (5). Gibbs and Ford studied other professions, primarily medical, legal, and engineering professions to see what aspects the professions
had in common. After their study of existing professions, Gary and Norman developed the infrastructure level model of a profession with eight components (5).

In my research, I found that considering only the infrastructure level components create a potential flaw in the analysis of a profession’s maturity. For instance, the knowledge component of the practitioner level plays a major role in the maturity of the profession because it has a significant effect on multiple infrastructure level components. I will explain the definition and relevance of knowledge in detail in section 3.10 below.

Table 3-1 - The components of a profession.

<table>
<thead>
<tr>
<th>Practitioner Level Components</th>
<th>Infrastructure Level Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professionals (people)</td>
<td>Initial Professional Education</td>
</tr>
<tr>
<td>Knowledge</td>
<td>Accreditation</td>
</tr>
<tr>
<td>Professional Practice (activities)</td>
<td>Skills Development</td>
</tr>
<tr>
<td></td>
<td>Certification</td>
</tr>
<tr>
<td></td>
<td>Licensing</td>
</tr>
<tr>
<td></td>
<td>Professional Development</td>
</tr>
<tr>
<td></td>
<td>Code of Ethics</td>
</tr>
<tr>
<td></td>
<td>Professional Society</td>
</tr>
<tr>
<td></td>
<td>Body of Knowledge</td>
</tr>
</tbody>
</table>
Figure 2 shows the interactions between the eight infrastructure level components of a profession (5). Many of the components affect professional practice, and almost all of the components affect other components in some way. Figure 2 uses arrows to show that one component affects another, and is labeled with what sort of relationship one component has on the other.

In the following sections, I will explain how I will measure the maturity of each component of the model when I apply it to software engineering. Then, for each component, I will describe what defines the component and what would qualify the
component as being *mature*. I will also relate each component with other mature professions to validate its inclusion into the maturity model.

3.1. Qualifying the Evolutionary Stages

To measure the maturity of each of the components of a mature profession, a simple, qualitative scale described in Table 3-2 is used. This scale measures the level of each component from being nonexistent to maturing (5).

<table>
<thead>
<tr>
<th>Table 3-2 - Evolutionary stages to characterize the infrastructure-level components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Nonexistent</td>
</tr>
<tr>
<td>2. Ad Hoc</td>
</tr>
<tr>
<td>3. Specific / Established²</td>
</tr>
<tr>
<td>4. Maturing</td>
</tr>
</tbody>
</table>

3.2. Initial Professional Education

For my purposes, I will use the U.S. Code of Federal Regulations definition of professional education, which is the same definition used in the Gibbs model: Education that provides “knowledge of an advanced type in a field of science or learning customarily acquired by a prolonged course of specialized intellectual instruction and study” (45).

Initial professional education for common professions such as engineering, law, and pharmaceutical occur at the baccalaureate and post-baccalaureate level (5). Examples of professions and the typical entry–level degrees include (5):

² In the original model, the term ‘specific’ is used to designate when a component exists. I will use the term ‘established’ instead since it is clearer.
In many professions, there can also be a broad range of concentrations within the profession, resulting in a wider range of initial professional education. For instance, Baylor University offers seven concentrations in their Doctor of Jurisprudence program: administrative practice, business litigation, business transactions, criminal practice, estate planning, general civil litigation, and intellectual property (46). This shows how more diverse disciplines require a variety of initial professional education programs.

Initial professional education can even change over time for a given profession. For example, in the middle of the century, engineering undergraduate degrees were commonly five-year programs (5). Over time, universities were pressured into evolving these degrees into four year programs. However, the recent rapid growth in technology is making it increasingly difficult for schools to maintain a four year program (5). Many people believe that students should instead pursue a five-year engineering program which would lead to both a bachelor’s degree and a master’s degree (47; 48).

<table>
<thead>
<tr>
<th>Medicine</th>
<th>Doctor of Medicine (MD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician</td>
<td>Doctor of Dental Surgery (DDS)</td>
</tr>
<tr>
<td>Dentist</td>
<td>Bachelor of Science in Nursing (BSN)</td>
</tr>
<tr>
<td>Nurse</td>
<td></td>
</tr>
<tr>
<td>Law</td>
<td>Doctor of Jurisprudence (JD)</td>
</tr>
<tr>
<td>Lawyer</td>
<td></td>
</tr>
<tr>
<td>Engineering</td>
<td>Bachelor of Science in Civil Engineering (BSCE)</td>
</tr>
<tr>
<td>Civil Engineer</td>
<td>Bachelor of Science in Mechanical Engineering (BSME)</td>
</tr>
<tr>
<td>Mechanical Engineer</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Bachelor of Architecture (BArch)</td>
</tr>
<tr>
<td>Architect</td>
<td>Bachelor of Landscape Architecture (BLA)</td>
</tr>
<tr>
<td>Landscape Architect</td>
<td></td>
</tr>
<tr>
<td>Accounting</td>
<td>Bachelor of Science in Accounting</td>
</tr>
<tr>
<td>Accountant</td>
<td></td>
</tr>
</tbody>
</table>
3.3. Accreditation of Professional Education Programs

The types of initial education within a profession can vary significantly in material and emphasis between schools. Accreditation provides quality assurance of education of a particular curriculum or program (49). In the U.S., there are two major types of accreditation bodies; bodies that accredit public and private schools, colleges, and universities within a region of the U.S., and there are bodies that accredit specific programs (50).

There are many specialized accreditation bodies that accredit individual programs within a school. These bodies are considered legitimate if they are recognized by the Council on Recognition of Postsecondary Accreditation (CORPA) and by the U.S. Department of Education (51). Examples of accrediting bodies for common professions include:

- Accreditation Board for Engineering and Technology (ABET) (49)
- Accreditation Council for Pharmacy Education (52)
- Accreditation Review Commission on Education for the Physician Assistant (53)
- American Bar Association (54)
- American Dental Association (55)
- American Optometric Association (56)
- American Society of Landscape Architects (57)
- National Architectural Accrediting Board (58)

3.4. Skills Development

Historically, the most common form of skills development took place during apprenticeships. It wasn’t until 1916 that over 50% of practicing professional engineers had any type of college degree (5). Now, much of skills development is learned during initial professional education. While the purpose of education is to teach knowledge of a
profession, courses tend to introduce skills development through laboratory work, semester-long projects, and design competitions\(^3\) (5; 47).

Skills can also be developed during on-the-job apprenticeships such as cooperative education work experience and internships. These skills development activities generally range from as short as eight weeks to up to six months, and typically occur before completion of initial professional education. These apprenticeships allow students and young professionals to develop skills in a professional environment before fully entering professional practice (59).

Although skills are expected to be acquired prior to starting professional practice, skills can and are developed during early professional practice (5). Recent engineering graduates spend time as an engineer in training prior to taking a licensing exam and becoming a registered professional engineer (14).

### 3.5. Certification

Certification and licensing are often confused with each other because the purpose of each is to ensure the competence of professionals. However, the two are not exactly the same. Informally, the differences between licensing and certification are that licensing is mandatory and regulated by the government whereas certification is optional and is usually regulated by a professional society (60). Both certification and licensing attest to the quality and reliability of an engineer, but the optional aspect of certification means that safety-critical projects are not guaranteed to have a certified professional software engineer in charge of the project (14).

\(^3\) Cal Poly SLO emphasizes a “Learn by doing” philosophy where students learn primarily through labs and other hands-on activities (38).
There exist many different types of certifications. Certification programs are sometimes administered by a professional society, sometimes by not-for-profit organizations, and some certification programs are offered by commercial companies relating to their own products and/or services. The latter, certification offered by commercial companies, is not considered professional certification because it covers specific commercial tools and/or services, not the best practices or methods in the profession.

For a couple of examples, certification exists for accounting and medical professionals. One widely known certification program is the Certified Public Accountant (61). The medical profession has an extensive certification program with specializations in over 20 different areas of medicine (62). This shows how many different professions have certifications of their profession, and in some cases, for specific specializations within their profession.

3.6. Licensing

As described above, licensing is a mandatory process administered by the government. Licensing standards for a profession are defined and maintained by the profession itself (10). Specific licensing laws and regulations are handled by the state; however, most states tend to adhere to national organizations who advise them on licensing requirements and exam content (5).

The purpose behind licensure is to provide assurance of the quality of a professional practitioner in order to protect public health, safety, and welfare (10). For instance, The Texas Engineering Practice Act states its purpose is to (63):

“Protect the public health, safety, and welfare; enable the state and the public to identify persons authorized to practice engineering in this state;"
and fix responsibility for work done or services or acts performed in the practice of engineering.”

Two of the most widely recognized professions are medicine and law (51).

Engineering is also a licensed profession. The laws on what kind of work can only be performed by a licensed engineer vary from state to state. However, most states exempt engineers in industrial corporations from the licensing requirements. For example, section 6747 of the California business and professions code states (64):

“This chapter, except for those provisions that apply to civil engineers and civil engineering, shall not apply to the performance of engineering work by a manufacturing, mining, public utility, research and development, or other industrial corporation, or by employees of that corporation, provided that work is in connection with, or incidental to, the products, systems, or services of that corporation or its affiliates.”

To clarify, this code specifies that this chapter on professional engineers does not apply in certain cases. These cases are defined by engineering work by employees working for an industrial corporation when the work is a product or service of that corporation. More simply, this means that engineers working for a company on a company product are exempt from licensing statutes.

As a result of this and other similar state laws across the nation, many engineers do not seek licensure. Individuals can work as engineers as long as they are working for a company under the above definition or as an employee under a registered professional engineer (10). Critical systems must have a licensed engineer on the project to sign off on it, though (10). In these cases, the registered professional engineer or the company is responsible for the product, not the individual employees. Table 3-4 shows the percentage of various engineers who seek licensure.

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Licensed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civil</td>
<td>44</td>
</tr>
</tbody>
</table>
Professional development includes all activities intended to improve or maintain the currency of the knowledge and skills of a profession after starting professional practice (5). This includes anything from reading a professional magazine to partaking in continuing education or training.

Since professional development has such a wide breadth of things it covers, it wasn’t possible to find consistent examples across the range of common professions (5). There are, however, two patterns amongst professions.

First, professional development is most important in professions who have a rapidly evolving body of technological knowledge on which their professional practice is based (5). Medicine, for instance, constantly acquires new knowledge about the genetic basis of diseases, new pharmaceuticals, and new diagnostic and treatment technologies (5). This places a high demand on physicians to stay up with current knowledge in their field.

Second, professional development tends to focus on small activities with short term gains for particular projects, rather than on long-term career development (5). For example, it is more common for professionals to take a short course on a specific tool or technique that will be used in their next job assignment than it is for them to take courses on more fundamental advances in their fields (5).
3.8. Code of Ethics

In order to ensure that practitioners behave in a responsible manner, professions develop and adopt a code of ethics and professional practice. A code of ethics is a statement by the profession, for the profession (65). This acceptance of and adherence to a code of ethics even makes many individuals feel that they are a part of a community of professionals (5). The creation and maintenance of a code of ethics demonstrates a profession taking its responsibility to the public seriously. This is further demonstrated by the profession’s specifications of sanctions or discipline for members who violate the code.

Having and maintaining ethics is important because the public is entrusting their lives and safety to certain professionals such as doctors and lawyers. Adhering to a code of ethics and professional practice assigns responsibility to the professional and provides assurance that the professional will act ethically within his ethical obligations. A code of ethics is also used as the basis for legally holding a professional responsible for malpractice mistakes (12).

3.9. Professional Society

In general, as a profession develops, voluntary associations of professionals tend to emerge. These may start as a scholarly society whose purpose is to promote an exchange of knowledge for professional practice. These associations can grow into organizations with a wide range of goals and responsibilities and develop into a defined professional society (5).

Activities of a professional society include publishing journals, conducting conferences or symposiums, designing model curricula for professional education
programs, and publishing text or reference books for professionals (5). They also take on regulatory functions such as defining certain criteria and managing certification programs, accreditation standards. They define a code of ethics and disciplinary actions for violating such codes. Some societies take on political roles such as lobbying legislative bodies and engage in litigation in matters of concern to the profession. Most professions have several associative societies that are a particular branch or specialization within the profession. Computer professionals are represented by organizations such as ACM, IEEE-CS, ICCP, and others (66; 67; 68).

3.10. Body of Knowledge

A body of knowledge is not a part of Ford and Gibb’s original maturity model. They could not come up with a way to define when a body of knowledge matures, so they only use the infrastructure level components in their model (5). In my research, I find a body of knowledge to have a crucial impact on the maturity model. Therefore as one of my contributions, I portray the body of knowledge as an important aspect in characterizing the maturity of a profession. A body of knowledge describes the complete set of concepts, terms, and activities that a professional discipline consists of (33). Usually, this includes a large amount of overlap into many other bodies of knowledge. For instance, a profession that commonly manages other individuals could have an overlap into a management or systems engineering body of knowledge.

The body of knowledge for a profession is important to consider in its maturity because it significantly affects multiple infrastructure level components of a profession. A professional society can take part in developing a body of knowledge. The body of knowledge itself influences accreditation and initial professional education by
establishing what knowledge is necessary for competent professionals (31). A body of knowledge also influences certification and licensing by outlining the minimal knowledge a professional must know in order to perform safe, reliable, and quality practice. This helps in outlining what material a certification or licensing exam should cover (10).

There are cases where a body of knowledge is constantly changing at a rapid pace. This is most notable in software engineering and medicine (69; 5). To accommodate this, many states, for medicine especially, require continuing education to maintain current knowledge and understanding in the field (70). Since this is only a standard, not all states adhere to the continuing education requirement of registered professionals (69). This is an issue the Texas Board wants to address because continuing education can at least aid a professional in at least knowing “when they don’t know” (71).

Software engineering has a guide to its body of knowledge (72). Software engineering is also fairly unique in this because while professions have a body of knowledge, they typically are not displayed so explicitly (72). Since the publishing of the guide to a software engineering body of knowledge, other professions have developed a guide to their own body of knowledge, including civil engineering and environmental engineering (73; 74). A body of knowledge can be defined explicitly for a new profession to accelerate its maturation rather than wait for natural growth of the profession (75). So for a young profession, it is important to consider the body of knowledge in characterizing the profession’s maturity level.

The guide is not a body of knowledge itself. Rather, it is meant to describe what portion of the body of knowledge is generally accepted and to provide topical access to it.
The body of knowledge itself exists in the published literature and common practices for a profession and isn’t necessarily tangible (72).

### 3.11. Summary of the Maturity Model of a Profession

Common professions have nearly all of the components in this model. Medical professions have all of these components. Architecture, accounting, and engineering professions exhibit seven of the eight characteristics (5). They each have either certification or licensing, but generally not both. For common professions, nearly all components are in the maturing stage, meaning components have been in place many years, during which they have come under active stewardship of an appropriate body in the profession and are continually improved upon (5). Gibbs makes a note that the professions in which practitioners are self-employed and offer their professional services directly to the public seem to have somewhat more mature components than professions in which the practitioners tend to be employed in large organizations. It is also important to note that due to the wide variation in the forms these components take, this model is primarily descriptive and not prescriptive (5).

This model can still be used with some confidence to predict the evolution of the infrastructure of components of emerging professions. Ford and Gibbs projected what the evolution of software engineering would look like in Table 3-5. This table shows at what point each component of the model will reach the appropriate stage. I added a rating for the maturity of a body of knowledge in Table 3-6. Even though Ford and Gibbs believe that we can’t measure when a body of knowledge matures due to its constant evolution, section 3.11 shows that we can measure the maturity by when a body of knowledge can be recognized and agreed upon (5). In the following chapter, for each component, I will
use these tables as a rubric to measure the state of software engineering at selected intervals.

Table 3-5 – Projected evolution of software engineering as a profession.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Professional Education</td>
<td>Bachelor’s degrees in computer science, engineering, mathematics, etc. are the common preparation for entry into the profession.</td>
<td>A recognized form of initial professional education in software engineering exists, but there is no standard curriculum.</td>
<td>Curricula reflect the best practice; nationally accepted model curricula exist; model curricula are regularly reviewed and revised.</td>
</tr>
<tr>
<td>Accreditation of Education</td>
<td>Accreditation based on computer science or engineering criteria.</td>
<td>Accreditation based on software engineering criteria; ABET and CSAB merged.</td>
<td>Accreditation guidelines are regularly reviewed and revised.</td>
</tr>
<tr>
<td>Skills Development</td>
<td>Some student project work in schools; some co-op programs; some company training programs for new hires.</td>
<td>Guidelines have emerged for the skills needed by a software engineer for entry into the profession.</td>
<td>Skills development mechanisms are in place and widely used (such as apprenticeships or engineer-in-training programs); skills for distinct specializations are recognized and developed.</td>
</tr>
<tr>
<td>Certification</td>
<td>ICCP, ASQC certification; commercial certification related to software packages and technologies.</td>
<td>Certification as a software engineer; nationally recognized certification standards.</td>
<td>Certification in specialty areas within software engineering; nationally recognized specialty certification standards.</td>
</tr>
<tr>
<td>Licensing</td>
<td>State licensing as a professional engineer under existing statutes.</td>
<td>Some state licensing examinations address software engineering skills specifically.</td>
<td>Licensing is based on appropriate examinations; NSPE and NCEES collaboration; recognized as protecting the public in appropriate situations.</td>
</tr>
<tr>
<td>Professional Development</td>
<td>Individuals pursue professional development as they determine the need.</td>
<td>Professional development guidelines (curricula expenditures per year, etc.) have</td>
<td>Recognized generalist and specialist career paths for software engineers; nationally recognized education and training guidelines and curricula.</td>
</tr>
</tbody>
</table>
Table 3-6 - Rubric for measuring maturity of the body of knowledge.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code of Ethics</td>
<td>Codes of ethics of ACM, IEEE, ASQC, ICCP, engineer licensing statutes.</td>
<td>Code of ethics specifically for software engineers.</td>
<td>Code widely respected and adopted; the profession has mechanisms to discipline violators.</td>
</tr>
<tr>
<td>Professional Society</td>
<td>ACM, IEEE Computer Society, others.</td>
<td>Society explicitly states that it represents software engineering.</td>
<td>Society has appropriate range of products and services for software engineers.</td>
</tr>
<tr>
<td>Body of Knowledge</td>
<td>Bodies of knowledge exist for IT professionals and other computing professionals.</td>
<td>A body of knowledge specific to software engineering is recognized.</td>
<td>The body of knowledge is widely accepted and used for developing education curricula, licensing and certification standards, and more.</td>
</tr>
</tbody>
</table>


Since I will be using this maturity model to determine whether or not licensing is appropriate for software engineering, I need to define what level of maturity is required before licensing is acceptable. These criteria can be subjective, so I will use the maturity of electrical engineering at the time that it acquired a standardized licensing exam as a basis. Since it is difficult to determine exact states of each maturity component at specific times in history, I will consider whether a component is established or not.

Electrical engineering was targeted for licensure at the same time as all of the other engineering disciplines (8). Therefore, it was licensed at the same time and had a standardized PE licensing exam at the same time as other disciplines including civil and mechanical engineering; in 1966 (9). Electrical engineering, along with all of the other engineering disciplines, did not have certification or licensing in place at this time, which
was the purpose of developing such an exam for all engineering disciplines (8). Therefore, these two components were not established.

According to ABET’s website of accredited programs, there were 142 accredited electrical engineering programs in the United States by 1965 with the first accredited program receiving accreditation in 1936 (43). The requirement for having a discipline-specific FE exam is 100 accredited undergraduate programs nationwide (76). Therefore, we can conclude that electrical engineering had established programs for initial professional education as well as established accreditation standards to accredit such programs. This also means there were many years worth of practice and knowledge from which a body of knowledge would become established.

Electrical engineering was initially represented by two professional societies; the American Institute of Electrical Engineers (AIEE) and the Institute of Radio Engineers (IRE) (77). AIEE formed in 1884, when electricity was first becoming a major force in society, and IRE was founded in 1912 (77). The IRE modeled itself after AIEE, but focused on radios. It expanded to include electronics. These two societies merged in 1963 to form the IEEE (77). When it formed, IEEE had 150,000 members, 140,000 of which were from the U.S. (77). This shows that electrical engineering had an established professional society that held a large, well-established community before the first EE PE exam was developed.

The first code of ethics adopted by an industry group was actually adopted by the AIEE in 1912 (78). The American Society of Civil Engineers followed suit shortly after in 1914. Therefore, not only was a code of ethics established for electrical engineering, but electrical engineering was the lead into industry groups adopting a code of ethics.
I was unable to find distinct examples prior to 1966 that electrical engineering had established forms of skills and professional development. However, as seen by how well-established all of the other areas were by this time, I believe it is reasonable to assume that there was some form of skills and professional development for electrical engineers. Before merging, AIEE and IRE had conferences and periodicals, which are parts of these two components (77).

Based on the research of the maturity of electrical engineering, all other areas besides licensing and certification were established before it became licensed. But the engineering disciplines became licensed based upon the need of proficiency and assurance in professional work, not based upon the maturity of the discipline itself (8).
4. The Maturity Model Applied to Software Engineering

Chapter 3 provides a qualitative maturity model for professions. In this chapter, I will apply this model to software engineering, using the rubric at the end of chapter 3 to measure which stage each component is in. Gilda Pour followed Gibbs’s analysis in 2000 with a survey using this same model (79). Within Steve McConnell’s book published in 2004, Professional Software Development, the author discusses the profession of software engineering and includes an updated maturity level with Gibbs’s maturity model (25). I’ll report my findings on the maturity level of each component in 1996, 2000, 2004, and 2010. I will not strictly use the results derived from these previous papers, but these works from Gibbs, Pour, and McConnell will assist me in making accurate measurements of the maturity model at each time frame I am reviewing. The progression through these years allows us to see how each component has developed and matured over time.

At the end of this chapter, I will summarize my findings on all components from 1996 to 2010. From this summary, I will answer if software engineering has become mature enough to warrant licensure.

4.1. Initial Professional Education

For review, the rubric for initial professional education criteria is shown in Table 4-1.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Professional Education</td>
<td>Bachelor’s degrees in computer science, engineering, mathematics, etc. are</td>
<td>A recognized form of initial professional education in software engineering exists, but</td>
<td>Curricula reflect the best practice; nationally accepted model curricula exist;</td>
</tr>
</tbody>
</table>
### 4.1.1. 1996 – Initial Professional Education

Practitioners in industry from before 1970 started in other fields such as electrical engineering, physics, or math and became software engineers (5). Many of the more recent practitioners have degrees in computer science or computer engineering or by contrast do not even have a college degree or other post-baccalaureate degree (5). In this initial phase of the profession, there were many variations in existing computer science programs. While software engineering specific programs have started appearing in other countries, they are only recently beginning development in the United States. In 1996, uniform initial professional education did not exist for software engineers (5).

Initial education accepted for software engineers meets the *ad hoc* criteria set out in the rubric, “Bachelor’s degrees in computer science, engineering, mathematics, etc. are the common preparation for entry into the profession.” Therefore, since initial education exists in the forms of degrees in computer science, math, and other disciplines, initial professional education was in the *ad hoc* stage in 1996 (5).

### 4.1.2. 2000 – Initial Professional Education

IEEE-CS and ACM formed The Software Engineering Education Project (SWEEP) in 1998. SWEEP provides a detailed set of guidelines for software engineering programs that will eventually seek accreditation. SWECC officially adopted the SWEEP accreditation guideline in Dec 1998. Also in 1998, the Computer Science Accreditation Board (CSAB) started a formal integration with ABET, a merger that will unify the
criteria and process used to accredit software engineering programs. SWEEP will use the new accreditation guidelines as a specification to design one or more model curricula for software engineering, leveraging the prior work from SWEBOK, the Computer Curriculum 2001 task force, and others.

There are an increasing number of software engineering BS degrees offered by computer science departments and new software engineering departments in the U.S., the U.K., Europe, Canada, and Australia. While some controversy exists over the differences between software engineering education and computer science education, one aspect to all good software engineering programs is the recognition that software engineering is fundamentally different from both computer science and computer engineering (79). These differences were described earlier in section 2.2.1. Making this distinction of software engineering programs from computer science and computer engineering allows the program to focus on attributes most important to software engineering practices (23).

With initial education curriculum standards being created and software engineering programs starting to be offered across the nation, this component is nearing the established stage. It hasn’t reached that point because software engineering isn’t yet recognized as the common preparation for entry into the profession, as the rubric requires. Therefore, this component is in the ad hoc stage according to the rubric, but is evolving to the established stage.

4.1.3. 2004 – Initial Professional Education

In the spring of 2001, members of the Software Engineering Education Project (SWEEP) began planning work for the Software Engineering Volume (31). In the fall of 2001, SWEEP was replaced by the joint IEEE-CS and ACM Computer Curriculum
Software Engineering (CCSE) Steering Committee. This Committee was formulating the final Software Engineering curriculum guidelines, including information specific to the two-year college environment in 2004 (31).

Bachelor’s degrees exist in computer science, electrical engineering, mathematics, and other disciplines which were considered common degrees for entry into the software engineering profession, meeting the rubric criteria for being *ad hoc* (25). For software engineering, dozens of master’s degree programs and many new bachelor’s degree programs exist; however, few students have completed any of the undergraduate software engineering programs and a software engineering degree isn’t seen as common entry requirements into the profession (25). Therefore, this component would be considered *ad hoc* in 2004, progressing into the *established* stage.

**4.1.4. 2010 – Initial Professional Education**

In 2004, The Joint Task Force on Computing Curricula formed by the IEEE Computer Society and the Association for Computing Machinery, released curriculum guidelines for undergraduate degree programs in software engineering (31). These guidelines establish core requirements for an undergraduate curriculum. As core units, these are units that IEEE-CS and ACM determined as necessary coursework for a software engineering student. The areas covered in the core are (31):

- Computing Essentials
- Mathematical and Engineering Fundamentals
- Professional Practice
- Software Modeling and Analysis
- Software Design
- Software Verification and Validation
- Software Evolution
- Software Process
- Software Quality
• Software Management
• Systems and Application Specialties

The core curriculum requirements were designed to be minimal to leave as much room as possible for tailoring elective components (31). This allows curriculum to meet individual needs of the institution offering them.

More software engineering programs have developed since 2004. There are currently 31 Bachelors of Science programs in the U.S. (80). In addition, 21 of those programs are ABET accredited, including Cal Poly SLO (43).

With software engineering program standards, we have “a recognized form of initial professional education in software engineering [that] exists” (31). This is the requirement for this component to be in the established stage in accordance with our rubric.

These standards have been developed by IEEE-CS & ACM to reflect best practice. This curriculum is regulated by the two organizations nationally recognized as representing software engineering; IEEE-CS and ACM. Therefore, this meets the first two criteria of the maturing stage of our rubric; “Curricula reflect the best practice; nationally accepted model curricula exist.” However, as of yet, there is no indication that this is regularly reviewed and revised, which is the third criteria in our rubric for the maturing stage. The closest we have is evidence that as of summer 2010, planning for Computing Curricula 2013: Computer Science volume has started (81). Therefore, this component is in the established stage.

4.1.5. Summary – Initial Professional Education

Initial professional education started out in the ad hoc stage, but has been maturing into the established stage it is in now. It has been established for multiple years
now and the only feature preventing this from entering the mature stage is regular review and revising of this component. Table 4-2 shows how initial professional education has matured since 1996.

Table 4-2 - Summary of maturity progress for initial professional education.

<table>
<thead>
<tr>
<th></th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Hoc</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Established</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Accreditation

For review, the rubric for accreditation criteria is shown in Table 4-3.

Table 4-3 - Rubric for accreditation.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accreditation of Education</td>
<td>Accreditation based on computer science or engineering criteria.</td>
<td>Accreditation based on software engineering criteria; ABET and CSAB merged.</td>
<td>Accreditation guidelines are regularly reviewed and revised.</td>
</tr>
</tbody>
</table>

4.2.1. 1996 – Accreditation

There are two accreditation bodies who may have jurisdiction over accreditation of software engineering programs, The Computer Science Accreditation Board (CSAB) and the Accreditation Board of Engineering and Technology (ABET). CSAB has jurisdiction over accreditation of computer science programs, and ABET has jurisdiction over accreditation of engineering programs (82; 49). In 1996, neither of these bodies had accreditation standards for software engineering. With the similarities existing between software engineering and computer science, it is expected that there will be conflict between CSAB and ABET over who will have jurisdiction for accreditation of software engineering (5). Fortunately, the two accreditation bodies plan on merging (5). This will prevent a territorial battle over who has jurisdiction as well as will allow experts in
computer science and experts in engineering to coordinate cooperatively towards the
development of acceptable accreditation standards.

With no accreditation standards or singular accreditation body specifically for
software engineering, this category only meets the criteria for the *ad hoc* stage of our
rubric; there only exists “accreditation based on computer science or engineering
criteria.”

### 4.2.2. 2000 – Accreditation

In December 1998, CSAB started formal integration with ABET (82). CSAB
accredits computer science programs and ABET accredits engineering programs. This
merger will unify their accreditation criteria and processes used to accredit software
engineering programs (79). Once this merger is completed, uniform accreditation for
software engineers can be established. The merger is not yet complete, so accreditation is
still in the *ad hoc* stage, but will soon evolve into the *established* stage.

### 4.2.3. 2004 – Accreditation

CSAB and ABET started their formal integration in December 1998. The
mechanics of the integration were resolved in the early 2000s with the transition
completing in 2001 (82). The two accreditation bodies are now merged and have
established standards for accrediting software engineering. Auburn University has the
first accredited software engineering program, receiving its accreditation in 2002 (43).
This meets the requirements of the *established* stage, “Accreditation based on software
engineering criteria; ABET and CSAB merged,” putting accreditation in the *established*
stage.
4.2.4. 2010 – Accreditation

With CSAB and ABET merged and accreditation standards for software engineering in place, accreditation is clearly established for software engineering. The first program to become accredited received its approval in 2002 (43). In 2003, there were 15 of 23 software engineering bachelor programs with ABET accreditation (83). Now, there are 21 programs out of 31 that are accredited (43)(80).

In November 2008, the Computing Accreditation Commission approved proposed changes to the software engineering accreditation criteria and presented the proposal to ABET in October, 2009 (84). The board accepted comments for consideration until April 1, 2010. The criteria is scheduled for being adopted into criteria in the fall of 2010 and will be applied for accreditation actions during the 2011-2012 academic year.

From this we show that accreditation meets the established rubric criteria by accrediting programs based on software engineering criteria, and accreditation meets the rubric criteria for maturing because the “accreditation guidelines are regularly reviewed and revised.” Therefore, accreditation has recently developed into the maturing stage.

4.2.5. Summary – Accreditation

Table 4-4 shows how accreditation has developed quickly since 1996. The two accreditation bodies, CSAB and ABET have merged, accreditation standards for software engineering have become established, and have been subjected to review and revision since becoming established.

<table>
<thead>
<tr>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ad Hoc</td>
<td>1 Ad Hoc</td>
<td>2 Established</td>
<td>3 Mature</td>
</tr>
</tbody>
</table>

Table 4-4 - Summary of maturity progress for accreditation.
4.3. Skills Development

For review, the rubric for skills development criteria is shown in Table 4-5.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skills Development</td>
<td>Some student project work in schools; some co-op programs; some company training programs for new hires.</td>
<td>Guidelines have emerged for the skills needed by a software engineer for entry into the profession.</td>
<td>Skills development mechanisms are in place and widely used (such as apprenticeships or engineer-in-training programs); skills for distinct specializations are recognized and developed.</td>
</tr>
</tbody>
</table>

4.3.1. 1996 – Skills Development

Skills development requires developing skills in applying knowledge acquired during initial profession education. The first issue is that the skills required for a software engineering professional have not been defined. A few of the skills that Gibbs and Ford hypothesized would be useful are manual skills such as typing; structured behaviors such as those used during software technical reviews; and communication skills, which would include interviewing customers to elicit software requirements (5).

Initial professional education provides some opportunities for skills development. Skills can be learned through programming assignments in classes, during semester long projects, or by participating in co-op programs working with industry (85). Other skills are developed once a person begins his first job as a software engineer in industry (85).

This shows that current skills development is done through student project work in skills, some co-op and internship programs, and company trainings for new hires; the rubric criteria for *ad hoc*. Since the entry level software engineer skills guidelines haven’t emerged for this component to reach the *established* stage in the rubric, this category is in the *ad hoc* stage.
4.3.2. 2000 – Skills Development

Software engineering has many foci and requires different training for diverse software roles and specialties. These specialties include, but are not limited to, architect, system engineer, design engineer, test engineer, quality engineer, maintenance engineer, programmer, and technician, but these specialties aren’t standardized across the profession (79). These different problem domains require different specialty skills such as the skills needed for user interface design versus skills required for database or operating system development (79). These skills still aren’t defined, so skills development has not reached the established stage. Although the skills required aren’t defined, skills development still exists in the forms of student project work, co-op and internship programs, and training for new hires (5). This meets the rubric criteria for the ad hoc stage.

4.3.3. 2004 – Skills Development

Guidelines have been developed for the skills needed by a software engineer for entry into the profession (25). Some skill requirements are necessarily tailored to the specific entry job a company is offering, but a core set of skills required of a professional software engineer can be derived from these (25). For instance, software engineers will need at least a couple of years of experience in multiple programming languages. An engineer must also be capable of estimating about how long a given project should take (86). It is also important to understand how to elicit and trace back to requirements, handle configuration management tools, and similar responsibilities (87).
With guidelines having emerged for the skill requirements of an entry-level software engineer, the skills development component meets the rubric criteria for the established stage.

**4.3.4. 2010 – Skills Development**

Most companies have some form of internship or co-op opportunities for students to work and gain skills experience while attaining their education. However, after completing their education, most students simply enter the work force and begin practicing. There are not any ‘Engineer in Training’ or apprenticeship opportunities available for young professionals.

Additionally, it is accepted that there are specific specializations of software engineers, but these specializations and their skill sets are not all clearly defined. Some professionals enter specifically for system security, graphics, user interfaces, or another area. Though there are similarities, the skills required by a company seeking these individuals vary from company to company. Sometimes different types of software engineers are identified through another aspect defining what level of the software they work with, whether as a front-end engineer, back-end engineer, software manager, or a software developer (86).

This demonstrates that skills development is still in the established stage. Guidelines have emerged for the skills needed for an entry software engineer, which meets the rubric’s standards for being established. But core skills for distinct specializations are not fully recognized as developed, which the rubric requires for being in the maturing stage. Furthermore, the rubric requires some form of engineer in training or apprenticeship to qualify for the maturing stage.
4.3.5. Summary – Skills Development

The skills required for a software engineer entering the profession have been defined. In contrast, while there are also many specializations, but they are not distinctly defined and recognized, yet. Additionally, there are no established programs for engineer in training or apprenticeships, though this form of training is required for licensure and will follow licensure (14). It logically follows that establishing licensing can allow skills development to reach the mature stage.

<table>
<thead>
<tr>
<th>Year</th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ad Hoc</td>
<td>Ad Hoc</td>
<td>Established</td>
<td>Established</td>
</tr>
</tbody>
</table>

4.4. Certification

For review, the rubric for certification criteria is shown in Table 4-7.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certification</td>
<td>ICCP, ASQC certification; commercial certification related to software packages and technologies.</td>
<td>Certification as a software engineer; nationally recognized certification standards.</td>
<td>Certification in specialty areas within software engineering; nationally recognized specialty certification standards.</td>
</tr>
</tbody>
</table>

4.4.1. 1996 – Certification

Certification is administered in three ways: organizations specifically created for the certification, professional societies offering the certification, and commercial companies providing certification for their tools (5). Several forms of certification exist for software engineers. The best known is certification through the Institute for Certification of Computing Professionals (ICCP) which certifies information technology professionals (68). The most recent certification offered is the American Society for
Quality Control (ASQC) which certifies software quality professionals (5; 88). There are also commercial certifications through Novell, Apple, Microsoft, but these aren’t professional certifications in the sense that the professional maturity model represents.

The rubric criteria for established is “Certification as a software engineer; nationally recognized certification standards.” None of these certifications address the broad range of knowledge and skills needed by a software engineer. The only criteria that exists are those laid out in the ad hoc criteria of the rubric, “ICCP, ASQC certification; commercial certification related to software packages and technologies.” Therefore this component is in the ad hoc stage.

4.4.2. 2000 – Certification

The forms of certification available are limited and inconsistent offerings for tool or application-specific certifications. The most appropriate professional certification for software engineers is ICCP certification (68). However, ICCP certifies information technology professionals, not software engineering professionals. Since certifications of computing professionals and certification software packages and technologies exist, but they are not clearly defined for software professionals, this component fits the rubric criteria for the ad hoc stage but not the established stage.

Gilda places this component in her paper in the nonexistent to ad hoc stage because certification offerings were inconsistent, limited, and technology-based (79). However, based on the criteria we’ve established, this qualified for ad hoc.

4.4.3. 2004 – Certification

IEEE-CS established a Certified Software Development Professional (CSDP) certification intended for mid-level software engineering professionals (89; 25). IEEE
made this certification available to software practitioners in 2002 (89). Becoming certified through the CSDP requires passing an exam.

Eligibility for taking the CSDP is based off of education and experience. However, if you are a Senior Member of the IEEE or are a licensed software engineer, these requirements are waived (90). The education and experience requirements are (90):

- Education (at least one of)
  - You have a bachelor’s degree
  - You are an educator at the post-baccalaureate level
  - You are a full member of the IEEE
- Experience (at least one of)
  - You have an advanced degree in software engineering and at least two years (about 3,500 hours) of experience in software engineering/development.
  - You have at least four years (about 7,000 hours) experience in software engineering/development

These eligibility requirements are similar but less restrictive than licensing requirements. Licensing requires holding a bachelor’s degree from an accredited program whereas the CSDP exam does not require that the program held accreditation (14). Both require four years of experience in the field, but licensing has stricter requirements on being trained to practice as a professional during these four years (14).

The content on the exam provides assurance that the software engineer is competent in the fields covered in the software engineering body of knowledge as well as having knowledge in safety, security, and related disciplines (89).

With the development of the CSDP certification, the established criteria for certification, “Certification as a software engineer; nationally recognized certification standards” has been reached. This component is now in the established stage.
4.4.4. 2010 – Certification

The CSDP certification has been out for eight years. IEEE-CS has complemented this certification with a Certified Software Development Associate (CSDA) certification (91). IEEE-CS added being a CSDA certificate holder to the education eligibility criteria for taking the CSDP exam (90). There are no eligibility requirements for taking the CSDA, but the following are recommended.

- Recent software or computer engineering graduates
- Undergraduates in their final year of a bachelor’s degree program in software or computer engineering.
- Non-degree professionals with more than 2 years of programming experience

Meanwhile, there is no certification for specialty areas for a professional software engineer, which is required for this component to reach the maturing stage of the rubric. Therefore, certification is in the established stage since we have “certification as a software engineer.”

4.4.5. Summary – Certification

There are many certifications for various tools, products, and even for various types of computer professionals, but only the CSDP certifies individuals as software engineering professionals (90). Until certifications are offered for specific specializations of software engineering and these certifications are regularly reviewed and revised to accommodate for new technologies, certification will remain in the established state.

<table>
<thead>
<tr>
<th>Year</th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Hoc</td>
<td>1</td>
<td>1</td>
<td>2 Established</td>
<td>2 Established</td>
</tr>
</tbody>
</table>

4.5. Licensing

For review, the rubric for licensing criteria is shown in Table 4-9.
Table 4-9 - Rubric for licensing.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licensing</td>
<td>State licensing as a professional engineer under existing statutes.</td>
<td>Some state licensing examinations address software engineering skills specifically.</td>
<td>Licensing is based on appropriate examinations; NSPE and NCEES collaboration; recognized as protecting the public in appropriate situations.</td>
</tr>
</tbody>
</table>

4.5.1. 1996 – Licensing

There has been some pressure in various states for some form of licensure of software engineers, but nothing significant has come from it. In 1990, New Jersey had a bill which proposed licensing or regulation of software engineers. The bill passed the New Jersey general assembly, but received scrutiny at the state senate and was not adopted (5). Additionally, the bill’s wording was changed to replace the word engineer with designer, so even if the bill had passed, it likely would have been moot (5). Therefore, no form of licensing exists specifically for software engineers (4).

Despite the lack of licensure specifically for software engineers, software engineers can still be licensed under existing statutes, however, there is little incentive to do so. Most states waive licensure requirements for those working in industry (4). Furthermore, the knowledge required to acquire a license under current statutes would not be productive towards education to become a quality software engineer (4; 5).

Since no specific means of licensure exists for software engineers, this component has not reached the *established* stage in the rubric. However, software engineers can be licensed as a “professional engineer under existing statutes,” so licensing is in the *ad hoc* stage.
4.5.2. 2000 – Licensing

Texas has requested that ACM and IEEE-CS establish standards for licensing software engineers. These professional societies agreed to this to ensure that software engineering was properly understood when licensing statutes were put in place for software engineers. In 1998, Texas started licensing software engineers (4). Licensing has also been approved in the UK and some regions and territories of Canada (79).

Since Texas does not have an exam to license software engineers, the state instituted an exam-waiver clause to allow software engineers to become licensed. This waiver allows an individual to become licensed if they have either 16 years of experience if they hold a bachelor’s degree or 12 years of experience if their bachelor’s degree is from an accredited program (92).

This makes Texas the only state to license software engineers; however, since it isn’t available in all other states, and no states have any licensing exam for software engineers, this component fails to meet any of the criteria for being established. Software engineers, even in Texas, must be licensed “as a professional engineer under existing statutes” as the rubric states for ad hoc. Therefore, this component is still in the ad hoc stage.

Gilda places this component in the ad hoc to established stage because it exists in Texas and other countries (79). However, my scope is on only the United States, and since Texas does not offer an exam for licensing, this component does not meet the criteria for established.
4.5.3. 2004 – Licensing

Texas started licensing software engineers in 1998 (4). Texas has continued licensing software engineers by either having them take a non-software engineering exam or by expecting that they meet eligibility requirements for an exam-waiver (92). Texas doesn’t have any available exam for software engineers and a new nationally accepted exam can’t be approved by NCEES standards until at least 10 states request such an exam (10). Since this hasn’t happened yet, licensing can’t meet the requirements for reaching the established stage and is still in an ad hoc stage.

4.5.4. 2010 – Licensing

In August 2009, NCEES approved IEEE to move forward with the development of a PE exam for software engineers. IEEE-USA is the lead technical society sponsoring the examination with cooperation from other organizations including IEEE-CS and NSPE (93).

The software engineering PE exam will be independent of the electrical engineering and computer engineering PE exams. Although the specifications for the exam are not yet developed, the registration committee predicts that there will be no more than 20% overlap between the computer engineering and the software engineering PE exams (93).

While laws are state specific, NCEES has an existing model law recommendation for licensure of engineering professionals which requires (10):

1. A four year ABET accredited degree in the appropriate discipline
2. Successful completion of an FE exam
3. Verifiable and documented evidence of four years of engineering experience
4. Completion of a PE exam
Previously, software engineers could not become licensed because the infrastructure wasn’t in place to meet this model law. Currently, there are 21 accredited software engineering programs; there is a general FE exam that any engineer could take; and engineering experience is gained while working in the profession (43; 10). Therefore, the development of a PE exam is all that a software engineer needs now to meet these requirements.

Once the PE exam is finished, the established level of the licensing component will be reached; “Some state licensing examinations address software engineering skills specifically.” This shows that licensing is close to reaching an established stage. Until this happens, though, licensing will remain in the ad hoc stage.

### 4.5.5. Summary – Licensing

Licensing has not matured at all on our model since 1996. Texas is the only state that licenses software engineers, but Texas doesn’t even have an exam for software engineers to take. However, IEEE-USA and IEEE-CS have been approved to develop such an exam and are in the process of doing so. After this exam is developed, validated, and approved, states can start adopting this exam. Then, licensing will become established and can move into the established stage on our model. This is expected to happen within the next few years.

<table>
<thead>
<tr>
<th></th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Hoc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.6. Professional Development

For review, the rubric for professional development criteria is shown in Table 4-11.
<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional Development</td>
<td>Individuals pursue professional development as they determine the need.</td>
<td>Professional development guidelines (curricula expenditures per year, etc.) have emerged.</td>
<td>Recognized generalist and specialist career paths for software engineers; nationally recognized education and training guidelines and curricula.</td>
</tr>
</tbody>
</table>

**4.6.1. 1996 – Professional Development**

Continuing education and training directly related to software engineering are widely available. This includes individual courses and graduate programs in universities, professional development courses, conferences offered by professional societies, courses provided by training vendors, and in house programs in large software companies (5). Software engineers seek out these professional development opportunities as they see the need, which meets the criteria for this component being *ad hoc* on the rubric. One problem still remains before this component can become established, though; No standards or guidelines for professional development of software engineers have emerged, which is required for this component to reach the *established* stage of the rubric (5). Therefore, this is in the *ad hoc* stage.

Ford and Gibbs reported this component as being *established* because continuing education and training exist directly related to software engineering and are widely available (5). I am being more conservative with my maturity ratings, though, and maintain that it is still in the *ad hoc* stage in 1996 because it doesn’t fully meet the *established* criteria; software engineering doesn’t meet having professional development guidelines available.
4.6.2. 2000 – Professional Development

Several masters programs exist for software engineering professional development. In fact, there are more master’s programs then there are bachelor’s programs (79). There are also fragmented course offerings, extensions courses, seminars, professional conferences, and manufacturer certification programs opportunities available to software engineers wishing to pursue professional development (79). But guidelines for professional development still have not emerged to raise this component to the established stage of the rubric, so professional development is still in the ad hoc stage.

4.6.3. 2004 – Professional Development

Some organizations have published professional development guidelines (25). Construx, for example, proposes a ladder model to provide guidance and support for career advancement. Construx is based off of the SWEBOK. It introduces four capability levels from introductory to mastery to qualify how developed a software professional is in a given skill or knowledge area (94). Using a matrix of skill level and knowledge level, Construx evaluates current professional development and potential growth based on the level of mastery in knowledge and experience of a particular area (94). The established criteria for professional in the rubric states “professional development guidelines have emerged” and the emergence of these guidelines provides measurable growth for a software engineer.

With all the professional development opportunities and established guidelines for professional development, this component is in the established stage. McConnell reports this as being in the ad hoc stage and progressing into the established stage due to the reasons I provided in this section (25). This difference in rating could be attributed to
McConnell being more conservative than I am with his ratings, or possibly due to changes occurring in 2004, when his book was published. Either way, we both agree in general on how mature this component of software engineering is in 2004.

4.6.4. 2010 – Professional Development

Many forms of professional development exist for software engineers including continuing education, master’s programs, and conferences (5; 25). Software engineers pursue professional development as needed, but companies also encourage their employees to pursue professional development. Professional development guidelines for software engineers have been proposed by companies such as Construx (94). The guidelines proposed by Construx provide a way to measure the growth and level of professional development an individual has reached. They also outline how the individual can pursue further professional development, and whether it needs to be in knowledge, experience, or both (94). This meets the rubric criteria for professional development to be in the established stage.

Professional development guidelines have not been created or recognized for specialist career paths, which is required for professional development to progress into the maturing stage. Therefore, this component is still in the established stage.

4.6.5. Summary – Professional Development

Guidelines have been created for professional development in software engineering. Many masters programs exist; more than there are bachelor’s programs for software engineering. ACM and IEEE-CS host or are a part of numerous conferences. This among many other professional development activities clearly put this component in the established stage. For this component to reach the mature stage specializations of
software engineering need to be clearly defined and guidelines for these specializations need to be clearly defined.

Table 4-12 - Summary of maturity progress for professional development.

<table>
<thead>
<tr>
<th></th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Hoc</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

4.7. Code of Ethics

For review, the rubric for code of ethics criteria is shown in Table 4-13.

Table 4-13 - Rubric for code of ethics.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code of Ethics</td>
<td>Codes of ethics of ACM, IEEE, ASQC, ICCP, engineer licensing statutes.</td>
<td>Code of ethics specifically for software engineers.</td>
<td>Code widely respected and adopted; the profession has mechanisms to discipline violators.</td>
</tr>
</tbody>
</table>

4.7.1. 1996 – Code of Ethics

In 1993, ACM and IEE-CS formed a joint task force to improve the profession of software engineering. One of the joint task force’s goals is to establish a code of ethics and professional practice on which to document the professional and ethical responsibilities and obligations of software engineers (95). Dr. Don Gotterbarn is the chair of the committee currently working on this code of ethics. This step, once met, will meet the requirements to reach the established stage of the code of ethics component; “code of ethics specifically for software engineers.”

Many computer professional codes of ethics exist, but none of these specifically address the software engineering profession. An accepted code of ethics is usually considered a prerequisite to both certification and licensing (5). An example of a code of ethics is the ICCP code of ethics, but this code is for information technology professionals, not software engineering professionals (68). The two societies
professionalizing the software engineering profession, IEEE-CS and ACM, each have their own code of ethics (66). But again, neither of these codes of ethics is applied specifically to software engineers. The existence of a code of ethics for computing professionals such as IEEE-CS, ACM, and ICCP meet the criteria for this component to be in the *ad hoc* stage in accordance with the rubric.

Because a code of ethics specific to software engineers does not exist at this time, this component is in the *ad hoc* stage.

### 4.7.2. 2000 – Code of Ethics

The Software Engineering Coordinating Committee (SWECC) established and released the Software Engineering Code of Ethics and Professional Practice Version 5.2 in 1998 (96). This code of ethics is only recently developed, so it is not widely known or practiced yet.

There also are not any formal consequences for professionals who violate the code (79). In other professions, a licensing board hears accusations and can recommend disbarment (as in the legal profession) or some other similar professional sanction. Such a code could also be used in legal proceedings to determine whether or not a software engineer has acted in accordance with professional norms and the accepted body of knowledge (5).

The code of ethics for software engineers is established which fits the *established* criteria in the rubric, but it is not widely known or practiced. Being widely known and practiced and having mechanisms to discipline violators of the code are the criteria required by the rubric to reach the *maturing* stage. There also are no formal sanctions for
engineers who break part of the code of ethics. Therefore, the code of ethics is in the *established* stage of maturity.

Gilda reports this component being between the *ad hoc* and the *established* stage because the code of ethics is established, but isn’t widely known or practiced (79). Despite this, the code of ethics component still meets the criteria in the rubric I am using for being *established*.

4.7.3. 2004 – Code of Ethics

The ACM and the IEEE Computer Society have adopted a code of ethics specifically for software engineers. The code is more widely known than when it was first released, but it still isn’t widely known and practiced (25). Some schools have ethics courses for software engineers to teach students how to act ethically and professionally (47). In order for a software engineering program to become ABET accredited, they must require their students to take a course on ethics and professional practice (42).

The code of ethics doesn’t meet the criteria outlined in the rubric for being *maturing* since the code is not “widely respected and adopted” and there do not exist “mechanisms to discipline violators of the code.” The code of ethics does meet the *established* requirements by specifically targeting software engineers. Therefore, it is still in the *established* stage. Initial professional education and accreditation will contribute to widespread awareness and adherence to this code of ethics to bring this component close to the *maturing* stage.

4.7.4. 2010 – Code of Ethics

During their joint effort, ACM and IEEE developed and established the Software Engineering Code of Ethics (SECOE) in 1998 (34). Due to accreditation criteria, initial
professional education requires that software engineering students learn the code of ethics and how it applies to obtain a software engineering degree (42). Establishing the code of ethics enables in part the establishment of licensure for software engineers. Likewise, licensure will enable a professional society to institute sanctions for engineers acting against the code of ethics (5).

The code of ethics meets half of the criteria to be maturing; “[The] code [is] widely respected and adopted.” However, until the profession is regulated through a mechanism such as licensing, the rest of the criteria to be maturing, “the profession [having] mechanisms to discipline violators,” can’t happen. So we conclude that the code of ethics is still in the established stage.

4.7.5. Summary – Code of Ethics

The ACM/IEEE-CS Joint Task Force developed and released the Software Engineering Code of Ethics and Professional Practice. This meets one of the criteria for software engineers to become licensed. Once software engineers become licensed, ACM and IEEE-CS can define the mechanisms that will be used to discipline violators of the code of ethics.

Table 4-14 - Summary of maturity progress for code of ethics.

<table>
<thead>
<tr>
<th>Year</th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage</td>
<td>Ad Hoc</td>
<td>Established</td>
<td>Established</td>
<td>Established</td>
</tr>
</tbody>
</table>

4.8. Professional Society

For review, the rubric for professional society criteria is shown in Table 4-15.

Table 4-15 - Rubric for professional society.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional Society</td>
<td>ACM, IEEE Computer Society, others.</td>
<td>Society explicitly states that it represents software engineering.</td>
<td>Society has appropriate range of products and services for software</td>
</tr>
</tbody>
</table>
4.8.1. 1996 – Professional Society

No professional society specific to software engineering exists. Many professional societies exist which relate to computing, such as the Data Processing Management Association and ICCP, but these societies do not target software engineering as a whole (5). However, ACM and IEEE-CS provide a wide range of products and services for software engineers. These two societies have also committed to co-operate on a long term effort to establish software engineering as a profession (31). Therefore, it looks like one or both of these two professions will become the dedicated society for software engineering professionals. The existence of many computing societies meets the criteria to be in the ad hoc stage. To reach the established stage, a society must explicitly state that it represents software engineering. Since no societies have explicitly stated that they represent software engineers, this component is in the ad hoc stage.

4.8.2. 2000 – Professional Society

IEEE-CS and ACM have formed a coordination committee to define software engineering as a profession (79). The Software Engineering Coordinating Committee (SWECC) membership is evenly divided between the ACM and the IEEE-CS. Some of the major projects that SWECC is working on include the Software Engineering Body of Knowledge, the Software Engineering Code of Ethics and Professional Practice, and the Software Engineering Education Project (79). By this point, IEEE-CS and ACM have demonstrated that together they are representing software engineering professionals and
they have explicitly stated that they represent software engineering, the criteria required for this component to be in the established stage.

With IEEE-CS and ACM identified as the societies representing software engineering, this component has reached the established stage. Gilda puts this component in the established to mature stage, but this component hasn’t met all of the rubric criteria to qualify for being mature (79).

4.8.3. 2004 – Professional Society

In addition to IEEE-CS and ACM, other professional societies such as the Software Engineering Institute have explicitly stated that they represent software engineering (25). These societies do not offer the full range of products and services needed to support software engineers as professionals (25). The rubric requires professional societies to offer the full range of products and services required of software engineers for this component to be in the maturing stage. ACM and IEEE-CS have worked for development in order to offer the products and services required by software engineering professionals. They currently offer a wide variety of conferences and special interest groups (97; 98).

With the existence of multiple professional societies representing software engineering, this component is established. By providing more products and services to software engineering professionals, this component is moving towards the maturing stage.

4.8.4. 2010 – Professional Society

IEEE-CS and ACM are the two largest societies representing software engineers. Both societies offer periodicals, conferences, and digital libraries (66; 67). Both offer
select software tools made available to students studying software engineering. IEEE-CS offers a certification exam and material to study and prepare for the exam (90). The two societies have explicitly stated that they represent software engineering and provide a wide range of products and services for software engineering professionals. This meets the criteria for being both *established* and *maturing*. With this, I conclude that the professional society component is in the *maturing* stage.

4.8.5. Summary – Professional Society

Multiple societies are in place which represent software engineering. IEEE-CS and ACM are the two most recognizable of these societies. IEEE-CS is dedicated as far as trying to license the profession whereas ACM stands firmly against licensing (4).

Table 4-16 - Summary of maturity progress for professional society.

<table>
<thead>
<tr>
<th></th>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ad Hoc</td>
<td>Established</td>
<td>Established</td>
<td>Mature</td>
</tr>
</tbody>
</table>

4.9. Body of Knowledge

For review, the rubric for body of knowledge criteria is shown in Table 4-17.

Table 4-17 - Rubric for body of knowledge.

<table>
<thead>
<tr>
<th>Component</th>
<th>Ad Hoc</th>
<th>Established</th>
<th>Maturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body of Knowledge</td>
<td>Bodies of knowledge exist for IT professionals and other computing professionals.</td>
<td>A body of knowledge specific to software engineering is recognized.</td>
<td>The body of knowledge is widely accepted and used for developing education curricula, licensing and certification standards, and more.</td>
</tr>
</tbody>
</table>

4.9.1. 1996 – Body of Knowledge

A body of knowledge does exist for software engineers since a body of knowledge is implied in the knowledge of the field (72). This component is important for software engineering because since the profession is young, a body of knowledge must be
defined explicitly to accelerate the profession into a mature state. One of the tasks that the joint task force of ACM and IEEE-CS sought to accomplish was establishing a body of knowledge (34). The body of knowledge task force performed a pilot survey, which concluded that the task force would not have sufficient resources to properly compile a software engineering body of knowledge (75). This caused the body of knowledge pursuit to be delayed.

Many pieces of a body of knowledge exist such as technologies and processes. But since a body of knowledge for software engineering could not be recognized by its professionals, this component fails to meet the criteria defined in the rubric to be established. Therefore, I find the body of knowledge to be in the ad hoc stage. Ford and Gibbs did not review the maturity of a body of knowledge in their paper because they did not see it as important to the overall maturity of the profession (5).

4.9.2. 2000 – Body of Knowledge

Since software engineering is a new profession, the explicit definition of a body of knowledge accelerates the profession into maturity faster than waiting for it to naturally evolve (79). However, getting practitioners to agree on a core body of knowledge is a key cornerstone in any discipline. The SWEBOK project identifies knowledge areas based on texts and software engineering programs (72). The committee is running public reviews and surveys to reach consensus on the content of the knowledge areas (79). Their goal is to categorize software engineering knowledge material as either core, advanced, or research level to determine what should be taught and known at various professional development levels (79).
The body of knowledge task force determined in a preliminary study that they wouldn’t have sufficient resources to compile a Software Engineering Body of Knowledge (SWEBOK). Due to this, the task force pursued the creation of a guide to the software engineering body of knowledge instead (75). This group was created in 1997. The first body of knowledge report in 1999 with expectations that consensus wouldn’t be reached for about four years (79).

A software engineering body of knowledge report has been released, showing that this component is developing into the established stage. But expectations are that consensus won’t be reached for at least another four years. This means that a body of knowledge is not yet recognized for software engineers, which is required by the rubric to be established. This means that the body of knowledge component is still in the ad-hoc stage.

Gilda states that this component is in the established to mature stage because the IEEE-CS/ACM task force has released their first body of knowledge report and expect consensus to take at least 4 years (79). I believe that this is a very poor rating for the body of knowledge given the reasons. First of all, the first report has just been released and there are many years that will pass before any consensus is even expected. This is far from being mature. Secondly, as there’s been no consensus and the report is the body of knowledge in progress, it hasn’t been released yet. Therefore, it’s too early to even claim that the body of knowledge is established.

4.9.3. 2004 – Body of Knowledge

ACM believes as of 1999 that defining a body of knowledge can’t be done yet (4). They’ve also decided to stop supporting tasks that will lead towards licensure of software
engineers. For these reasons, ACM has backed out of the pursuit to define a body of knowledge, leaving IEEE-CS as the sole professional society pursuing this task (4).

IEEE-CS completed defining and approved SWEBOK in 2004 (44). About 500 software engineers took part in the development of and approval of the SWEBOK (32). This makes the body of knowledge recognized for software engineers, qualifying this component for the *established* stage according to the rubric.

Even though this establishes a body of knowledge specific to software engineers, it is not fully agreed upon by software engineers who were not a part of its development (75). In fact, many software engineers who volunteered to create it expressed their opposition to releasing this as the software engineering body of knowledge (32).

A body of knowledge now exists and is in the *established* stage for software engineers. Unfortunately, there is unresolved and heated controversy over its development. This is a component not rated by McConnell (25).

### 4.9.4. 2010 – Body of Knowledge

IEEE-CS maintains the established SWEBOK. They are looking into updating the body to its third version. The changes they plan to incorporate are (33):

- A new knowledge area on Professional Practice
- Removal of three related disciplines: Computer Science, Mathematics, and Software Ergonomics – obviated by other changes
- Added material about Human-Computer Interfaces in the Software Design and Software Testing knowledge area
- Removal of the Software Tools section from Software Engineering Tools and Methods, and distributing it to the other knowledge areas
- A renamed Software Engineering Methods knowledge area to focus on methods that affect more than one knowledge area
- Redistribution of some other material into different knowledge areas
Proposed knowledge areas for new Guide to the SWEBOK are Measurement and Security (33).

The CSDA certification is already aligned with this revised outline for the SWEBOK. The CSDP is not, but will be revised to reflect these changes (33). It may be premature to make changes to the certification before completion of these proposed changes are in place.

IEEE-CS has an established body of knowledge for software engineers which they are continually updating to reflect currency of software engineering knowledge areas. But ACM still opposes this body of knowledge and believes that since the foundations of the project are faulty, any changes to the SWEBOK will continue to fail. Therefore, while the body of knowledge meets the rubric criteria to be established, the body of knowledge may still be inappropriate.

4.9.5. Summary – Body of Knowledge

A body of knowledge for software engineering is established and has been explicitly defined by IEEE-CS. However, the integrity of this body of knowledge and the quality assurance it promises are in conflict. While this component is in the established stage, it is so strongly opposed by highly respectable and knowledgeable software engineering professionals that it could arguably be in the ad hoc stage.

<table>
<thead>
<tr>
<th>1996</th>
<th>2000</th>
<th>2004</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ad Hoc</td>
<td>1 Ad Hoc</td>
<td>2 Established</td>
<td>2 Established</td>
</tr>
</tbody>
</table>

4.10. Summary of Software Engineering Maturity

Table 4-19 and Figure 3 show how the maturity levels of components have developed over time. Licensing remains as the only component that has not developed
into a more mature stage since Ford and Gibbs developed their model and is now the only component that is still in the *ad hoc* state.

Table 4-19 - Maturity level of each content area over time.

<table>
<thead>
<tr>
<th>Content Area</th>
<th>Maturity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1996</td>
</tr>
<tr>
<td>Initial Professional Education</td>
<td>1</td>
</tr>
<tr>
<td>Accreditation</td>
<td>1</td>
</tr>
<tr>
<td>Skills Development</td>
<td>1</td>
</tr>
<tr>
<td>Certification</td>
<td>1</td>
</tr>
<tr>
<td>Licensing</td>
<td>1</td>
</tr>
<tr>
<td>Professional Development</td>
<td>1</td>
</tr>
<tr>
<td>Code of Ethics</td>
<td>1</td>
</tr>
<tr>
<td>Professional Society</td>
<td>1</td>
</tr>
<tr>
<td>Body of Knowledge</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4-20 - Meaning of the maturity rating values.

<table>
<thead>
<tr>
<th>Maturity Rating</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nonexistent</td>
</tr>
<tr>
<td>1</td>
<td>Ad-Hoc</td>
</tr>
<tr>
<td>2</td>
<td>Established</td>
</tr>
<tr>
<td>3</td>
<td>Mature</td>
</tr>
</tbody>
</table>

Figure 3 - Maturity of the components over time.
Additionally, from the summaries provided in the above sections, we can determine that instituting licensure for software engineers will allow the skills development and code of ethics components to reach the mature stage. It will force the profession to add apprenticeships to meet the licensing requirements, which matures the state of skills development (5; 10). It will also give professional societies the ability to discipline violators of the code of ethics (5).

This would all go to imply that now is an appropriate and even an opportune time for licensing. Unfortunately, the software engineering profession still hasn’t resolved the case of conflicts with the SWEBOK and it being inappropriate for software engineers. Since the SWEBOK will be used to develop the content of the PE exam for licensure, and software engineers will be held to the standards and practices outlined in the SWEBOK, it is imperative that we ensure the body is appropriate before proceeding with developing an exam. Therefore, I believe that development of a body of knowledge formally accepted by the software engineering community is a necessary requirement and the last requirement before we can conclude that software engineering is ready for licensure and development of a PE exam.

4.11. Threats to Validity

Now I must consider potential threats to the validity of my work. Potential threats include the coverage of the material I researched, personal opinion and bias in assigning maturity ratings, the quality of a qualitative measure of the maturity of software engineering, and misconception about the body of knowledge.

In my research, I started by finding a few relevant papers and software professionals whose research areas are relevant to this thesis. I focused on the ACM and
IEEE Digital Libraries for general papers and lists of published papers on home pages of software professionals whose interests are in this domain such as Dave Parnas and Steve McConnell. Once I found the central papers for my thesis, I sought out current papers by using Google Scholar to find papers that have cited the papers I had been researching. Despite all of this, it is possible that I could have missed material on software engineering and professionalization that would have been relevant to analyzing its maturity.

For rating the different maturity levels of each component, I used a rubric to measure the components as qualitatively as possible. Even with a rubric, though, it is possible for personal bias and opinion to affect the rating. Therefore, I chose to only mark a component at a certain maturity level if it met all the requirements in that rubric for that maturity level. I also settled for marking components at one distinct stage that the component has clearly reached, even when previous papers would mark a component as being in a range of two stages (79; 25).

Finally, my conclusions rely significantly on my findings about the body of knowledge. I found a body of knowledge to be established, but due to disagreement over its validity, I find the answer to whether or not software engineers should be licensed difficult. I may be siding too much with the arguments against the body of knowledge or not considering the body of knowledge with enough weight.

Any of these three areas could affect the validity of my work. But having seen what the threats to my validity are, I have done my best to mitigate these threats.
5. Next Steps

Any component of the maturity model that hasn’t already reached the *mature* stage is a viable next step for maturing the profession. However, I am focusing on steps towards licensure of the profession. Therefore, next steps include work on the Principles and Practices of Engineering exam, the Fundamentals of Engineering exam, and on defining a software engineering body of knowledge.

5.1. Principle and Practices of Engineering Exam

The last major milestone for licensing software engineers is the development of a principles and practices of engineering (PE) exam for that discipline. In an IEEE-CS survey of software engineers in industry conducted on September 2008, 62.9% of respondents agreed that software engineers should be licensed if they practice in areas affecting public health, safety, and welfare. In addition, 61.5% supported development of software engineering licensure through NCEES Model Law (99). This shows that the majority of practicing software engineers is in support of licensure.

A minimum of 10 states need to request for NCEES to develop a PE exam before the development of a new exam can start (83). By 2009, 12 states have requested such an exam, meeting this requirement (99). IEEE-USA and IEEE-CS have been appointed to work jointly with NCEES and NSPE to develop such an examination (99).

IEEE-USA and IEEE-CS will have licensed engineers who practice software engineer produce specifications for the content of a software engineering licensure examination through surveys and meetings (93). Once the exam specifications are developed, IEEE will form a committee of software engineers to develop the exam

- 73 -
questions, to be approved by NCEES. After NCEES receives the committee’s software engineering PE exam, each individual licensing board will make a decision on whether or not to license software engineers in their state or territory (93).

5.2. Fundamentals of Engineering Exam

According to NCEES requirements, for an FE exam with a discipline-specific half to be created, there must be at least 100 accredited programs nationwide in that discipline (4)(76). The first program was accredited in 2002. By the start of the 2010 academic year, there were 21 accredited software engineering programs (43). Figure 4 shows how the total number of software engineering undergraduate programs which held accreditation each year. This graph shows how software engineering programs with accreditation quickly reached 13 after the third year. However, the number of accredited programs has been increasing at an average rate of only one or two programs per year since.

![Accredited SE Programs by Year](image)

Figure 4 - Number of ABET accredited software engineering programs by year.

I tried comparing this against how many software engineering programs existed across the United States. This proved to be difficult because the degree itself is a
relatively new degree compared to other engineering programs, and in many schools, it is still considered a specialization of computer science, computer engineering, or electrical engineering (24). According to Barron’s Profiles of American Colleges 2011, there are currently 31 software engineering programs in the United States (80).

Since software engineering evolved out of computer science, I decided to look at the number of computer science programs (23). There are over 200 accredited computer science programs nationwide (43). I have not looked into how many of these schools are engineering schools, but I believe that this demonstrates the potential that software engineering has to reach over 100 accredited programs nationwide. Many software engineering programs have developed out of computer science programs (23).

5.3. Body of Knowledge

IEEE developed the Software Engineering Body of Knowledge (SWEBOK) and it was agreed upon by many software engineers who had volunteered to assist in the project; approximately 500 software engineers in total (32). However, many respected software engineers and the ACM stand firmly against this body of knowledge. Among these are Dave Parnas, Cem Kaner, and Grady Booch (75).

The founders of the SWEBOK described the guide as a concept of generally accepted knowledge (100). Generally accepted knowledge applies to most projects most of the time, and widespread consensus validates its value and effectiveness (75). Generally accepted knowledge would imply that this knowledge should not be uniformly applied to all software engineering endeavors (75). However, the SWEBOK does imply that all software engineers should be equipped with this knowledge for potential implementation (33). The SWEBOK also defines generally accepted knowledge as
knowledge to be included in the study material of a software engineering licensing exam (33).

One issue with the body of knowledge is that it did not account for what is achievable good practice (32). The guide is too closely tied to textbooks that provide an inadequate view of software engineering because they are by definition targeted to a student audience. The SWEBOK also does not distinguish among potential roles within a software engineering project when discussing the body of knowledge that is required for that project (32). The SEWBOK doesn’t address knowledge requirements for different software engineering application domain areas. For these reasons and more, ACM believes that the initial SWEBOK development effort is flawed. This means that the process for updating the results will be based on a flawed effort, and therefore will be unlikely to succeed (32).

Cem Kaner points out that the SWEBOK unconditionally endorses IEEE Standard 829 for the software engineering test document, but no scientific research demonstrates that this is a good method, or better than others, or desirable under studied circumstances (101). This standard got put into the body of knowledge because it won a popularity contest among the volunteers to contribute to this project (101). What this does is put legal liability on the hands of professionals who believe in better or different practices than what SWEBOK constricts us to. In court, these professionals could be held liable simply for not following a standard in the SWEBOK, even if they used a standard more appropriate for the job that wasn’t listed in the SWEBOK (101).

ACM has expressed interest in developing a software engineering body of knowledge, but they believe that such a task is impossible as of 2000 (32). In order to
achieve a software engineering body of knowledge that is accepted and agreed upon by the professional software engineering community, ACM needs to reconsider their stance and take action in defining a more appropriate body of knowledge. This may mean defining specialized bodies of knowledge for specializations within software engineering since ACM believes that this is a more appropriate approach to take (32). On the other hand, if ACM is fully convinced that defining a body of knowledge is impossible at this time, they need to take it upon themselves to make this fact clear to licensing bodies and legal professionals. If we pursue licensure with an inappropriate body of knowledge, we will be held liable for standards that the professionals do not agree with. However, if ACM believes we don’t have a recognizable body of knowledge, then software engineers probably shouldn’t offer their services when there is potential to harm the public.
6. Conclusion

We can observe that software engineering has made significant steps in the past decade and a half in becoming more mature. Even though the profession itself isn’t as mature as other engineering professions and other types of professions, I have shown that licensing is the most immature component in the maturity model for software engineering. In addition, by establishing licensing for software engineers, other components such as the Code of Ethics and Skills Development can more easily reach the mature stage.

However, simply having all of the components of the model in place and at a mature stage isn’t enough. The components must be agreed upon and subscribed to by the practicing body of professionals to have meaning that is valuable to a profession’s practitioners. The Software Engineering Body of Knowledge is strongly opposed by many respected software engineers and by ACM. Therefore, until a body of knowledge is proposed that is agreed upon and accepted by the engineering community, software engineering is not mature enough for licensing. Once such a body of knowledge does emerge, we will not only be ready for licensing, but licensing will also lead to dramatically improving the maturity of the profession.
References


16. Licensing Boards. NCEES. [Online] NCEES.  


30. Parnas, Dave. *Do You Have a License to Drive That Mouse?*


42. ABET. *PROGRAM CRITERIA FOR SOFTWARE AND SIMILARLY NAMED ENGINEERING PROGRAMS.* s.l. : ABET, 2010.


60. Thompson, J. Barrie. It’s different on the other side of the pond. But should it be that way? s.l. : IEEE, 1999.


63. Texas Board of Professional Engineers. Texas Engineering Practice Act and Rules Concerning the Practice of Engineering and Professional Engineering Licensure. Austin, TX : s.n., 2009.

64. California Codes Business and Professions Code Section 6747. 2010.


102. NCEES Principles and Practice of Engineering CIVIL BREADTH and CONSTRUCTION DEPTH Exam Specifications. s.l. : NCEES.

103. NCEES Principles and Practice of Engineering Examination Electrical and Computer - COMPUTER Exam Specifications. s.l. : NCEES.


110. Leveson, Nancy and Knight, John. *Should Software Engineers Be Licensed?*


118. **Parnas, Dave.** Developing the Specifications and Questions for a Software Engineer PE Exam. [Email] 2010.


120. **Dominguez, Jorge.** The Curious Case of the CHAOS Report 2009. s.l. : Project Smart, 2009.

