

THE ASSOCIATION FOR COMPUTING MACHINERY AND LICENSING SOFTWARE ENGINEERS

Prentice Wongvibulsin

pwongvib@calpoly.edu

Professor Clark S. Turner

csturner@calpoly.edu

CSC 300 Professional Responsibilities

California Polytechnic State University

November 2008



- rev Cm -

This work is licensed under
Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>

Table of Contents

Introduction	5
Known Facts	6
The Association for Computing Machinery's Official Position	6
Software Engineering is Different	7
Software is More than Just Code	7
Concerns for the Public	8
Licensing in Texas	9
The Problem Comes with a Cost	10
What Licensing Provides	10
The Issue	11
The Arguments	11
Reasons Not to License	12
Reasons to License	15
Analysis	17
Texas as a Role Model	17
The Problem is Real: Overview of Safety-Critical Software Applications	18
Software Engineering Code of Ethics Analysis (SECOE 1.05)	21
Software Engineering Code of Ethics Analysis (SECOE 1.03)	21
Licensing is for the Public Good	22
Utilitarianism Perspective on ACMs Position Against Licensing	24
The Vision for Licensing Professional Engineers	24

Software Engineering Licensure Consortium	26
Software Engineering Code of Ethics Analysis (SECOE 1.08)	27
Deontological Perspective on ACMs Position Against Licensing	27
Public Awareness	28
Solving the Problem: Getting ACM Involved	28
Annotated Bibliography	30

Introduction

In 1907, Wyoming was the first state to pass a law requiring engineers to be licensed. Over forty years later in 1950, all the remaining 49 states introduced some form of licensing law for engineers¹. In 1998, the Texas Board of Professional Engineers adopted Software Engineering as a distinct discipline under which professional engineering (PE) licenses can be issued². That year, Texas approached the Association for Computing Machinery (ACM), the professional computing community of Software Engineers and other computer-related professions, requesting a licensure exam. ACM refused Texas' request by taking the position that Software Engineering is still too immature of a discipline to warrant licensing³. Currently, Texas is the only state which licenses Software Engineers. Will we have to wait forty more years for Software Engineering to become a licensed profession? Has Software Engineering matured to the point where it warrants licensing or will Software Engineering continue to be viewed as a non-licensable profession? The objective of this paper is to analyze ACM's position against licensing and determine if their decision is professionally responsible using various philosophies and the Software Engineering code of ethics (SECOE).

¹ What is licensure and why is it important? by NCEES <<http://www.engineeringlicense.com/licensure/>>

² Texas Board of Professional Engineers to License Software Engineers <<http://www.denveraitp.org/legislative/certification.html>>

³ Licensing of Software Engineers in Texas <<http://www.aitp.org/newsletter/2003julaug/article16.htm>>

Known Facts

The Association for Computing Machinery's Official Position

The Association for Computing Machinery (ACM) took the position in opposition of licensing Software Engineers as professional engineers in May of 1999. Their decision was based on the findings of the Software Engineering Coordinating Committee (SWECC)⁴. SWECC is a “permanent entity to foster the evolution of Software Engineering as a professional computing discipline.”⁵ SWECC started a project to identify a Software Engineering body of knowledge (SWEBOK) which can be used as a “generally accepted knowledge” for Software Engineering⁶. The Association for Computing Machinery (ACM) eventually withdrew from these efforts all together, stating that they felt “the activities of SWECC had become too closely associated with promoting the licensing of Software Engineers as Professional Engineers (PEs).” Association for Computing Machinery (ACM) believes that it is premature to license Software Engineers and that the licensing of Software Engineers would not effectively address the problems of software quality and reliability⁷. Licensing Software Engineers is demanded to assure public safety, but currently there is no test or exam to guarantee

⁴ ACM's Position on the Licensing of Software Engineers By John White and Barbara Simons (Publications of the ACM)

⁵ ACM's Position on the Licensing of Software Engineers By John White and Barbara Simons (Publications of the ACM)

⁶ SWEBOK: Project Overview

⁷ A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession (Final Version)

the competence of the candidate and determine that the candidate will not create software that will harm the public⁸.

Software Engineering is Different

The Association for Computing Machinery (ACM) makes another argument for their position against licensing that the framework provided by the National Society of Professional Engineers (NSPE) requires an applicant applying for a Software Engineering license to educate themselves in topics that do not require qualified Software Engineers. The most prevalent incompatibility with Software Engineering is the “general engineering” or Fundamentals of Engineering (FE) component,⁹ which is inappropriate for Software Engineers because it would prevent highly qualified Software Engineers from professional practice.¹⁰ The exam covers topics such as Strength of Materials, Engineering Dynamics (Statics and Dynamics), and Thermal Dynamics (*see* Fundamental Engineering (FE) Exam Specs <http://www.ncees.org/exams/fundamentals/fe_exam_specs.pdf> for the complete requirements).

Software is More than Just Code

“Good software must be well-engineered” states Dave Dorchester.¹¹ “A complex software system or program must be designed using engineering methods and thinking

⁸ Not Now, Not Like This. By Fran Allen, Paula Hawthorn, and Barbra Simons (Communications of the ACM)

⁹ Getting your P.E. License. By Reginaldo Alonzo Montague <<http://www.allbusiness.com/human-resources/employee-development/900934-1.html>>

¹⁰ A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession (Final Version)

¹¹ Why License Software Engineers? by Dave Dorchester. IEEE Software March/April 1999

to solve a problem.” It is important to clarify that this paper is discussing the licensing of Software Engineers (SE) and not mere code writers. The code writers are not qualified to perform Software Engineering tasks based on the skill set they have obtained as a programmer. A programmer learns skills such as how to code efficiently and usually knows a diverse set of languages.¹² On the other hand, we want to license a Software Engineer who is qualified to design and engineer software. A Software Engineer may be very well qualified to write code but his or her ability to make competent design decisions is more important. It must be clear to the licensing board that Software Engineers must have the ability to engineer those safety-critical systems to a specification that does not endanger public welfare. From here on, when we refer to a Software Engineer, we will be referring to the Software Engineer who works on safety-critical projects which have the potential to harm the public.

Concerns for the Public

Public health, safety, and welfare are threatened by software that lacks good engineering and testing as cases documented in many computer related accidents. Perhaps the most serious of those documented cases is the Therac-25 accidents which resulted in the deaths of six individuals.¹³ The Therac-25 is a medical linear accelerator used to destroy tumors with high-energy beams. The issue with the Therac-25 is that many of the hardware safety-checks, which existed in its predecessors, have been replaced by software. Consequently, a software bug caused the Therac-25 to

¹² Skills Needed in Programming. <<http://www.eskimo.com/~scs/cclass/progintro/sx1.html>>

¹³ An Investigation of the Therac-25 Accidents by Nancy Leveston and Clark S. Turner

administer excessive amounts of radiation and overdose the patient causing radiation poisoning and radiation burns. Another case where a software failure caused significant damage is the Ariane V incident which is possibly the most expensive¹⁴ documented software bug. The Ariane V is a European expendable launch system containing a software bug that caused a launch on July 4th, 1996 to end in failure after only about 40 seconds in flight.¹⁵ The rocket veered off course due to an improper cast from a 64-bit integer to a 16 bit integer, which was not large enough to hold the floating point value. This type conversion error and the failure of redundant inertial references systems (SRIs) led to the costly failure of the launch. Software faults caused by poor engineering which threaten public welfare can be mitigated by licensing professionals for these safety-critical projects

Licensing in Texas

Despite ACM's position against licensing, Texas has continued to license Software Engineers as professional engineers. This is their attempt to protect public health, safety, and welfare¹⁶ from poorly engineered software in safety-critical applications. Texas' decision has sparked the attention of many professionals to finally take action¹⁷ in addressing this long-standing issue in the software community. Texas' move to

¹⁴ The Most Expensive Bug. by Philippe Kaplan <<http://kabado-blog.blogspot.com/2008/03/most-expensive-bug.html>>

¹⁵ ARIANE 5: Flight 501 failure. Prof. J. L. Lions <<http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html>>

¹⁶ Why License Software Engineers? by Dave Dorchester (IEEE Software 1999)

¹⁷ Why License Software Engineers? by Dave Dorchester (IEEE Software 1999)

license Software Engineers has been attributed to the Therac-25 accidents which caused awareness of the need for a licensed Software Engineering professional.

The Problem Comes with a Cost

Failed systems have the potential to harm lives, waste taxpayer money, cause a lack of public services, cause security breaches, expose confidential data, and cause incorrect or missing information. The National Society of Professional Engineers estimated that \$275 billion is spent on software projects in the United States annually, \$60 billion of which is lost on failed projects.¹⁸

What Licensing Provides

The premise of licensing is to provide certain standards and levels of accountability for professionals who work on projects that could potentially “cause catastrophic harm to the public’s health, safety, and welfare” and mitigate “accidents” from occurring.¹⁹ Licensing happens at the state-level, meaning licensing laws vary from state to state. Licensing is required for engineers who provide engineering services directly to the public. The licensing requirement is the way in which public welfare is protected because it insures that a licensed individual has met certain requirements and has shown certain characteristics to indicate that he or she is competent and able to perform the service correctly and without harming the well-being of others. As computers (and their software) become increasingly relied upon by society for their correct function and

¹⁸ Is It Time to License Software Engineers? by Danielle Boykin. PE Magazine, December 2007.

¹⁹ Why License Software Engineers? by Dave Dorchester (IEEE Software 1999)

reliability, the Association for Computing Machinery's official position states that the Software Engineering discipline is not yet mature enough to be held to a standard which will protect public safety. Another mechanism of licensure is the ability to file a complaint against a Professional Engineer (PE). The following unethical acts violate the Professional Engineers Act: fraud, deceit, misrepresentation, negligence, incompetence, breach of contract, failure to use a written contract, or violating the Codes of Professional Conduct. Disciplinary actions can be taken such as suspension of license, revoking of license, placing licensees on probation, issuing administrative citations, and referring the matter to the district attorney for criminal prosecution.²⁰

The Issue

Should ACM support Software Engineering as a licensed profession?

The Arguments

Given the current state of this discipline and the urgent need for a form of regulation for safety-critical software, we must decide if ACM's position is ethical.

²⁰ Filing a Complaint against an Engineer or Land Surveyor or Unlicensed Person < http://www.pels.ca.gov/consumer/complaint_licensee.shtml >

Reasons Not to License

Why is the Association for Computing Machinery Against Licensing?

“[O]ur state of knowledge and practice in Software Engineering is too immature to warrant licensing. Moreover, Council felt licensing would be ineffective in providing assurances about software quality and reliability.”²¹ The Association for Computing Machinery (ACM) states in their official position that,

“the framework of a licensed professional engineer, originally developed for civil engineers, does not match the professional industrial practice of Software Engineering. Such licensing practices would give false assurances of competence even if the body of knowledge were mature; and would preclude many of the most qualified Software Engineers from becoming licensed.”²²

Even though ACM is opposed to the licensing of Software Engineers, ACM is committed to solving the problem of software quality by “promoting R&D, by developing a core body of knowledge for Software Engineering, and by identifying standards of practice.”²³

Software is Different

In other licensed engineering disciplines, the processes and standards for designing and building their products correctly is well understood and a design can be proven

²¹ A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession. ACM Publications.

²² A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession. ACM Publications.

²³ A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession. ACM Publications.

correct before being approved and constructed.²⁴ Software is different. No formal method or standard allows Software Engineers to prove that the software they have built is correct. In Software Engineering, the process of determining that the software performs its function as intended is known as verification. However, “verification [may] reduce program-testing load, [but] it cannot eliminate it.”²⁵ Furthermore, when software undergoes verification, test cases are often omitted even though these cases may have been identified as important. Sometimes test cases must be left out because they are impossible to check.²⁶ The mechanism used to reveal correctness is referred to as an oracle. In many cases, programs are considered “non-testable” because an oracle does not exist. For a trivial example, take a program that is to calculate the 900,000th digit of pi. An oracle for this problem does not exist because the 900,000th digit of pi is not known and thus an oracle can not be used to validate the result of this program. However, what is known as a partial oracle usually exists. A partial oracle allows the programmer to determine that a result is not the correct answer without actually knowing what the expected result should be. For example, when writing a program to determine the next largest prime number the expected result of the program is not known. We can not test this program by comparing the output to what we would expect it to be because the intention of the program is to calculate a value we do not know. Fortunately, we can test the validity of the output with a partial oracle. An example a few conditions for a partial oracle: verify that the next prime number is greater than the previous computed prime number and check that the result is really a prime number. If

²⁴ Lecture. Turner.

²⁵ No Silver Bullet: Essence and Accidents of Software Engineering. Frederick P. Brooks, Jr.

²⁶ On Testing Non-testable programs. Elaine J. Weyuker.

the result does not meet these conditions, we can confidently state that the result is incorrect.

This inability to guarantee or assure safety is the basis of an argument the Association for Computing Machinery gives against licensing.

“Is there a test that will assure the person who passes the test will be qualified to write programs that would never endanger the public? Will that person be qualified to sign off on program designs to assure that they are sound, just as building designs are signed by structural architects to assure the building is sound?”²⁷

Since there is no method to determine if a program is correct, there is no way a Software Engineer can determine that a program is sound. Because the process for verifying that a program is sound and will never endanger the public is currently unknown as well, it would be impossible to write an exam which would assess a person’s ability to write safe programs. “ACM believes that licensing would not be effective at providing assurances that Software Engineers could produce reliable, dependable, and usable software systems.”²⁸

²⁷ Not Now, Not Like This. By Fran Allen, Paula Hawthorn, and Barbra Simons. Viewpoint. Communications of the ACM Feb 2000.

²⁸ A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession. ACM Publications.

Reasons to License

Professional Engineers and the Test of Time

Paul Wongvibulsin, P.E., is shocked by the lack of accountability in engineering safety-critical systems. “Each day, millions of peoples’ lives depend on computer software reliability, which has major roles in many safety-critical applications in modern aerospace, automobile, and medical equipment industries. Therefore, Software Engineering should not remain unregulated,” says Paul, who has over thirty years of experience as a Mechanical Engineer in safety-critical projects. Paul has worked in safety-critical areas related to hydraulic flight control pumps and flight control actuators. “For many decades, the Professional Engineering Licensing has been proven in other engineering disciplines, such as Civil, Electrical and Mechanical Engineering, as an acceptable avenue to safeguard the life, health, property, and welfare of the public by regulating the practice of professional engineering.”

Many would agree with Paul’s statements although those in opposition to licensing often feel that the examination process does not fit the needs of Software Engineering licensure. Paul agrees that “[s]ome modifications to licensing requirements and regulations may be required to make them suitable and relevant to Software Engineering.”

Texas Paving the Way

The first engineers to become licensed were Civil Engineers. The licensing of Civil Engineers was sparked by the realization that unregulated design of construction projects such as bridges and roadways were a threat to public safety. Eventually the

laws were amended to cover other branches of engineering, such as mechanical and electrical engineering as well.²⁹ While Software Engineers produce a product that poses a threat to public safety as well, Software Engineering remains an unlicensed profession except in the state of Texas.

As of November 20th, 2008, querying the database of Professional Engineers (PEs) on the Texas Board (TBPE) Website (www.tbpe.state.tx.us) revealed 59 licensed professionals with a specialty in Software Engineering. According to Donald Bagert, who has met the majority of these licensed professional engineers, they have the background, experience and professionalism the computing community would expect of a Software Engineer³⁰. Frailey³¹ states that there are potentially two beneficial results of licensing Software Engineers:

- (1) Licensing forces us to define, codify, and organize our field and has the potential to improve the overall quality of the Software Engineering discipline.
- (2) Licensing may be legally required for certain purposes (very limited), it has the potential to reduce the chances that unqualified, incompetent practitioners will build inappropriate software in systems that affect the public's health and safety.

²⁹ A Brief History of the Board. California Board for Professional Engineers and Land Surveyors.

³⁰ Texas Licensing of Software Engineers: All's Quiet, For Now by Donald J. Bagert (Communications of the ACM, November 2002, Vol 45 No. 1)

³¹ Licensing Software Engineers by Dennis J. Frailey (ACM Viewpoint, Communications of the ACM, December 1999)

Analysis

To accomplish the goal of this paper and discover if ACM's position is ethical, we have taken a look at ACM's reasons for taking the position against licensing and the opposing reasons in support of licensing. We will take these arguments and determine if they hold up to the ethical standards presented in the Software Engineering Code of Ethics (SECOE) and if they hold up to deontological and utilitarianism philosophies.

Texas as a Role Model

Frailey's first statement that Texas is paving the way for advancements in Software Engineering as a discipline is already being proven true. With licensing in Texas, a big push has already been made to develop and improve Software Engineering as a discipline. Dorchester writes in "Why License Software Engineers?" that "progress is being made largely because the Texas Board decision added urgency to a long-standing problem in the software community."³² His second statement, that licensing has the potential to prevent unqualified individuals from working on safety critical projects may come true as well. Every other discipline to this date where a service is provided to the public and the service has potential to cause severe injury or damage,³³ has been regulated and licensed. Lance Kinney, P.E. states that "[i]t's very difficult now to become a licensed Software Engineer in Texas" because in 2006 they ended the

³² Why License Software Engineers? by Dave Dorchester (IEEE Software March/April 1999)

³³ Why License Software Engineers? by Dave Dorchester. Soapbox March/April 1999 IEEE Software

“experience-only path to licensure.”³⁴ In order to become a licensed Software Engineer, candidates can no longer skip the licensing exam by using their experience in the industry in place of examination. The licensing exam is still not perfect, but it certainly prevents those without any engineering knowledge at all from endangering the public with their incompetence. The belief that an inability to provide an assurance that one is qualified to practice Software Engineering without endangering the public is not sufficient reason to disregard it as an unacceptable solution to a very real problem.

The Problem is Real: Overview of Safety-Critical Software Applications

Safety-critical applications of Software Engineering are the primary concern when analyzing at the need for licensing. Failures in software can result in severe injury, death or extreme financial loss. The following is a brief overview of software in safety-critical environments derived from sections of Jacky’s paper on Safety-Critical Computing³⁵ and other sources^{36,37,38}:

In medicine: Software exists in just about all modern medical devices from small portable insulin pumps to large medical particle accelerators. The failure of software in

³⁴ Is It Time to License Software Engineers? by Danielle Boykin (National Society of Professional Engineers NSPE December 2007) PE Magazine, December 2007.

³⁵ Safety-Critical Computing: Hazards, Practices, Standards and Regulation by Jonathan Jacky <<http://staff.washington.edu/jon/safety-critical.html>>

³⁶ Bad software may have doomed Mars orbiter. MSNBC (Space.com). <<http://www.msnbc.msn.com/id/16580498/>>

³⁷ Illustrative Risks to the Public in the Use of Computer Systems and Related Technology by Peter G. Neumann ACM SIGSOFT (Jan 1994)

³⁸ Saab JAS 39 Gripen. <http://www.aeroflight.co.uk/types/sweden/saab/jas_39/gripen.htm>

these devices can prove fatal to their users. The Therac-25's life-threatening problems are an example of the hazards of software failure in medical device software.

In space: Spacecraft have millions of lines of code controlling their operation. A bug in this software can cause catastrophic failure and loss of life or incidents such as the Ariane V explosion. Another costly example of doomed spacecraft caused by faulty software is the Mars orbiter incident. According to NASA scientists, faulty software was uploaded to the spacecraft which caused the orbiter to expose an unprotected portion of itself toward the sun, effectively disabling the spacecraft.³⁹

In aviation: Aircraft are now being controlled by a fly-by-wire system, which use computer systems to control the aircraft. In August 1988, a Swedish JAS-39 fly-by-wire controlled aircraft crashed into trees after the controls did not respond to the pilot's commands to pull up.⁴⁰

In trains: Computer systems control switches which prevent trains from colliding.

In automobiles: Vehicles are using drive-by-wire systems which rely on computers to accelerate and break.

In power: Computers controlled equipment in factories and power plants (including nuclear reactors).

In weapons: Computers detect an attack, identify and track targets, aim guns and steer missiles, and arm and detonate explosives. Software is also used in computer systems which are responsible for defending targets. The Patriot Missile software problem is an example of such a computer system which contained a software bug

³⁹ Bad software may have doomed Mars orbiter. MSNBC (Space.com). <<http://www.msnbc.msn.com/id/16580498/>>

⁴⁰ Saab JAS 39 Gripen. <http://www.aeroflight.co.uk/types/sweden/saab/jas_39/gripen.htm>

resulting in the failure of the system and the death of 28 U.S. soldiers. The targeting system algorithm relied on a continuous integer value that became large as the system had been running longer. A precision error is introduced into the calculation from the from a floating point accuracy problem and as time increases, so does the inaccuracy. According to those who discovered the bug, after the missile system underwent 8 hours of continuous operation, the system suffered a 20% targeting inaccuracy. The incident causing the death of 28 soldiers and 98 injured occurred after the device had been running for over 100 consecutive hours without reboot even though the military knew about the software but, did nothing to fix it and encouraged the device to be rebooted every 8 hours.⁴¹

Neumann compiled a list⁴² of over 400 incidents where computer systems failed causing a loss of life, potential life-critical situation, loss of resources, a security problem, a system integrity problem, and a breach of privacy or other rights.

To ensure that a program would never endanger the public, a Software Engineer would have to show that bugs do not exist in the program. To find bugs, Software Engineers test their software. As we know from Weyuker's paper, "On Testing Non-testable Programs", testing shows the presence of bugs but not their absence⁴³. As a result of not being able to prove a program is bug-free, software in these safety-critical scenarios can never be guaranteed to be 100% safe.

⁴¹ Patriot Missile Software Problem. by Andrew Lum <http://www.cs.usyd.edu.au/~alum/patriot_bug.html>

⁴² Illustrative Risks to the Public in the Use of Computer Systems and Related Technology by Peter G. Neumann ACM SIGSOFT (Jan 1994)

⁴³ Lecture, Turner.

Software Engineering Code of Ethics Analysis (SECOE 1.05)

The Software Engineering code of ethics (SECOE 1.05) states that Software Engineers shall cooperate in efforts to address matters of grave public concern caused by software. As shown by the overview of software in safety-critical environments derived from Neumann's paper and the examples of several applications of software in safety-critical environments which can not be proven safe, the issue of regulating the construction of such software is a matter of grave public concern caused by software. The safety and welfare of the public is at hand and even though licensing may not be perfected yet, standing by and not helping to perfect the licensing framework is not the position ACM should be taking according to the Software Engineering code of ethics. The ethical course of action ACM should take is to assist Texas in developing their licensing exam.

Software Engineering Code of Ethics Analysis (SECOE 1.03)

The Software Engineering code of ethics (SECOE 1.03) states that a Software Engineer shall only approve software if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy, or harm the environment. The ultimate effect of the work should be to the public good. The fact that software is difficult to test and that we really have no way of proving that software is correct makes this rule very difficult. Fortunately, this is the same problem the licensure board is facing while trying to develop the exam for Software Engineers to become licensed. Because this is the code which Software

Engineers must abide by and the licensure board is attempting to accomplish the same thing, it is only logical to support the efforts of the board.

Licensing is for the Public Good

Licensing engineers has been around for over a century now with first Civil Engineer having been licensed in 1907. The process of licensing engineers has been relatively successful at its mission to “safeguard the life, health, property, and welfare of the public by regulating the practice of professional engineering”⁴⁴ in safety-critical projects. Licensing does not guarantee public safety or that the Software Engineer is going to produce perfect code but it substantially “reduces risks by carefully screening a candidate for licensure and providing some enhanced risk assurance that a PE is qualified by education, experience, or examination to design infrastructure that effectively protect the public safety.”⁴⁵

Becoming a licensed Professional Engineer (PE) is not a trivial task. To become a licensed Software Engineer (SE), the framework requires that the candidate (1) obtain a four-year degree from a university program with Accreditation Board for Engineering and Technology (ABET) accreditation. ; (2) Take and pass an eight-hour examination on the Fundamentals of Engineering (FE). Once the candidate has passed the Fundamentals of Engineering exam, he or she becomes known as an Engineering Intern (EI) or Engineer in Training (EIT)⁴⁶; (3) Passing the Fundamentals of Engineering

⁴⁴ Mission Statement. California Board for Professional Engineers and Land Surveyors. State of California. 2008.

⁴⁵ Interview with Dan Wittliff, P.E. Software Engineer by Sam Li

⁴⁶ How do I get licensed? NCEES. <<http://www.engineeringlicense.com/licensure/how.php>>

exam, the Engineering Intern or Engineer in Training is now allowed to gain experience which is required to become a Professional Engineer (PE)⁴⁷; (4) With the required years of experience working as an Engineering Intern or Engineer in Training and the proper recommendations from other Professional Engineers (PEs), the Engineering Intern or Engineer in Training is qualified to take the second and final exam to become a licensed professional. Once the Engineering Intern or Engineer in Training passes the Principles and Practice of Engineering (PE) exam, he or she is then given the distinguished P.E. designation.

These four steps require the candidate for licensure to obtain a good education, substantial relevant experience, and finally, prove that they have obtained the skills necessary to perform competently. The only concern is that the exam would not be able to accurately determine if the Software Engineer was in fact qualified to produce quality software. However, even though the process may not be perfect, having the software licensing in place provides the benefit of requiring Software Engineers to at least obtain a good education and have relevant experience. Also, it distinguishes those engineers who have undergone this process and gained valuable experience from those who have not undergone any sort of training and are less qualified to work on safety-critical projects. Perhaps the greatest motivation to push for the Software Engineering licensure is that more improvements can be made to it as the problems of licensing Software Engineers become apparent in practice rather than simply in theory.

⁴⁷ How do I get licensed? NCEES. <<http://www.engineeringlicense.com/licensure/how.php>>

⁴⁸Utilitarianism Perspective on ACMs Position Against Licensing

The utilitarianism perspective analyzes the situation by observing the consequences of the behavior or the action rather than solely the action itself. Furthermore, it determines the moral worth of an action by the overall utility of the outcome or results of the behavior or the action. In utilitarianism reasoning, an action is viewed as good when it produces happiness and an action is viewed as bad when it produces unhappiness.⁴⁹

Applying this philosophy to the topic of licensing and the decision to license Software Engineers, we reveal that licensing would provide the greatest good for the public by “safeguard[ing its] life, health, property, and welfare.”⁵⁰ The Association for Computing Machinery (ACM) opposes this by not helping build a framework which will essentially protect the public. Licensing Software Engineers will provide the public with the assurance that we are taking steps to protect them from poorly designed software. Licensure reduces the probability that an unqualified Software Engineer will work on a project and endanger the public causing unhappiness. The Association for Computing Machinery does not approve of this ethical choice and thus from a utilitarianism perspective that is unethical.

The Vision for Licensing Professional Engineers

The vision statement of the California Board for Professional Engineers and Land Surveyors states that “California Professional Engineers will possess the knowledge,

⁴⁸

⁴⁹ Computer Ethics. by Deborah G. Johnson 3rd Ed.

⁵⁰ Mission Statement. California Board for Professional Engineers and Land Surveyors. State of California. 2008.

skills, and abilities enabling them to meet the expectations of clients and consumers.”⁵¹

This is concurrent with what Software Engineers strive for and the missions of the various committees developing the Software Engineering discipline. Both the vision statement and the Engineers’ Creed share many parallels with the Software Engineering code of ethics (SECOE). With this said, it seems ACM and the licensure board have a common goal and there is no reason why ACM should not be working with licensure boards to develop and perfect the licensing process for Software Engineers.

Software Engineering Code of Ethics Analysis (SECOE 6.01)

The Software Engineering code of ethics (SECOE 6.01) encourages Software Engineers to organize and develop an environment favorable of acting ethically. By encouraging the licensing of Software Engineers, we are organizing an environment which requires us to act ethically. Furthermore, to become license as a Software Engineer, one must accept the Engineers’ Creed which states:

“As a Professional Engineer, I dedicate my professional knowledge and skill to the advancement and betterment of human welfare.

I Pledge:

To give the utmost performance;

To participate in none but honest enterprise;

To live and work according to the laws of man and the highest standards of professional conduct;

⁵¹ Vision Statement. California Board for Professional Engineers and Land Surveyors.

To place service before profit, the honor and standing of the profession before personal advantage, and the public welfare above all other considerations.

In humility and wit the need for Divine Guidance, I make this pledge.”⁵²

This creed has many similarities to the SECOE. For example, SECOE 6.05 which states that a Software Engineer is not to promote his or her own self interest at the expense of the profession, client or employer and the creed directly provides that service comes before profit and the profession before personal gain. SECOE 1.06 states that Software Engineers must be fair and avoid all deception in all statements, particularly public ones, concerning software or related documents, methods and tools.

Finally, licensing Software Engineers provides the best environment for encouraging engineers to act ethically because as a licensed engineer, a complaint can be filed against them and their license can be suspended, revoked, or they can be prosecuted. The decision to license Software Engineers or at least provide the opportunity for those who seek it is the ethical decision according to the Software Engineering code of ethics.

Software Engineering Licensure Consortium

In July 2007, The Software Engineering Licensure Consortium met to discuss the issue of building an examination for licensing Software Engineers. Representatives present at the consortium include those from the National Society of Professional Engineers’ (NSPE’s) Licensure & Qualifications for Practice Committee, NSPE’s Professional Engineers in Industry, the Institute of Electrical and Electronics Engineers-USA Licensing and Registration Committee, the IEEE Communications Society, the

⁵² Engineers’ Creed. From the interview with Dan Wittliff, PE, by Sam Li.

IEEE Computer Society, the Texas Board of Professional Engineers, the California Society of Professional Engineers, and several NSPE members who serve on state licensing boards. Notice anyone missing? The Association for Computing Machinery (ACM) did not appear at a consortium which was dealing with a topic directly related to how the industry will regulate its members.

Software Engineering Code of Ethics Analysis (SECOE 1.08)

The Software Engineering code of ethics (SECOE 1.08) encourages Software Engineers to volunteer professional skills to good causes and contribute to public education concerning the discipline. The Software Engineering Licensure Consortium is an example of how the Association for Computing Machinery can volunteer its professional skills to a good cause and contribute to public education of the discipline. A professional organization which expects to maintain a society of ethical Software Engineers sets a poor example from its decision to not support licensure and their resistance to the development of licensing processes.

Deontological Perspective on ACMs Position Against Licensing

The deontological philosophy views the rightness or wrongness of actions based on the rightness or wrongness of the actions themselves without considering the consequences of the actions.⁵³ The deontological analysis of the Association for Computing Machinery's (ACM's) decision observes the fact that ACM has chosen not to help license the field of Software Engineering and develop the discipline into a regulated

⁵³ Computer Ethics. by Deborah G. Johnson 3rd Ed.

environment. The Association for Computing Machinery has a duty to its community to look out for Software Engineers and its decisions must be made to ensure that the well-being of the public is not threatened because they are a professional community for this discipline. From a deontological perspective, ACMs position against licensing is unethical.

Public Awareness

With all the facts and arguments presented here, I do not doubt that we are at least concerned that safety-critical projects do not have licensed Software Engineers with the skills necessary to prevent injury or damage to the public. The concern is in two parts: (1) there are no Software Engineers that can be hired on to projects where a level of accountability is required and (2) there is no known process or method for any Software Engineer (currently) to deem a system completely safe and bug free. There was no founded documented cases where there was significant public⁵⁴ resistance to allowing a potentially unskilled, under-qualified Software Engineer from building a product that could one day cause their death. Raising public awareness increases the demand for Software Engineers to become licensed and will eventually encourage ACM to finally support licensure.

Solving the Problem: Getting ACM Involved

The Association for Computing Machinery is the professional body for Software Engineers, and not supporting licensing simply because they believe that the discipline

⁵⁴ By public I mean the non-engineering, non-technical public

is too immature is unethical and unacceptable. One of the two largest computing societies must encourage the community to work together to support this necessary and ethical choice to license Software Engineers.

Annotated Bibliography

What's in a Name? Tech Sector battles Engineers on "Software Engineering" by Mylene Sayo (Canada NewsWire)

< <http://www.peo.on.ca/enforcement/June112002newsrelease.html>>

This article talks about the argument that Software Engineers are not actually licensed engineers and should not be called engineers because most in fact do not even have a degree in Software Engineering but rather in computer science. This was not used in this paper because it refers specifically to Canadian licensed engineers.

Should We License Software Engineers? by John Knight and Nancy Leveson

< <http://www.cs.virginia.edu/~jck/publications/cacm.2002.pdf>>

They conclude that "licensing Software Engineers who work on safety-critical systems as PEs would neither be practical nor effective in achieving the goal of protecting the public interest, and it could even have serious negative ramifications."

National Society for Professional Engineers Licensure page

<<http://www.nspe.org/Licensure/index.html>>

This is simply a page that explains what licensure is and why one would go about obtaining it. This was relevant as research to what other engineers go through and the site explains the requirements of a PE. Furthermore, it provides a bit of history behind the emergence of licensed engineers.

SIGCHI Vol. 30 No. 1, January 1998 by Harry E. Blanchard (from Association of Computing Machinery Digital Library)

< <http://www.sigchi.org/bulletin/1998.1/standards.html?searchterm=medical+device+software>>

Speaks about standards and various institutions which are relevant to the development of medical device applications but not the theme of this paper.

Engineering Ethics: A Search For Solutions by Author E. Schwartz, NSPE General Counsel

< <http://www.nspe.org/Ethics/EthicsResources/Otherresources/searchforsolutions.html>>

Provides interesting guidelines and explains the necessity of an engineering code of ethics.

What is licensure and why is it important? by NCEES

< <http://www.engineeringlicense.com/licensure/>>

This article provides a history of licensure and how it emerged. In the United States, the first state to pass a bill requiring licensure of all engineers was Wyoming in 1907. By 1950 all states adopted licensing laws of some kind.

Failure Modes in Medical Device Software: An Analysis of 15 Years of Recall Data by Dolores R. Wallace and D. Richard Kuhn

< <http://csrc.nist.gov/staff/Kuhn/final-rqse.pdf>>

This paper should prove interesting once I actually read it. It has some statistics about data collected from medical device failures where no death occurred.

Open Source Everywhere by Thomas Goetz (Wired Magazine)

< <http://www.wired.com/wired/archive/11.11/opensource.html>>

This one talks about the benefits of open source software in medical device technology not only for safety but for productivity and cost savings.

General Principles of Software Validation; Final Guidance for Industry and FDA Staff by U.S. Department of Health and Human Services Food and Drug Administration

< http://www.fda.gov/cdrh/comp/guidance/938.html#_Toc517237953>

This is the document which provides guidelines for FDA approval on medical devices. I was hoping to find what is included in an FDA submission in here.

ACM: July 18, 2006 The Honorable Vernon Ehlers Chairman Committee on House Administration 1309 LHOB Washington DC 20515

This one compared the measures in place to ensure the reliability of medical devices systems vs. aviation systems vs. voting systems.

ACM: Correct by Design

< http://www.acm.org/ubiquity/interviews/pf/v5i2_poore.pdf>

UBIQUITY: As more computers get made, and more people who are not the best programmers, or even professional programmers at all, do this work, do you feel as if you're only putting your finger in the dike?

POORE: Yes, that sounds pretty accurate, unless there's some serious licensing so that only qualified people are allowed to work on devices that have implications on public safety such as transportation and **medical** devices, and serious certification of the devices themselves.

Licensing of Software Engineers in Texas

< <http://www.aitp.org/newsletter/2003julaug/article16.htm>>

This gives a brief summary of the licensing exam and requirements in Texas and what the ACM and other bodies have done.

ACM's Position on the Licensing of Software Engineers

By John White and Barbra Simons

Publications of the ACM

This article states ACM's position against licensing Software Engineers because they feel that the problems of software quality and reliability have not been addressed. Licensing would be viewed as a statement that SEs can produce software systems that are reliable and dependable.

Not Now, Not Like This

By Fran Allen, Paula Hawthorn, and Barbra Simons

Publications of the ACM - Viewpoint

This talks about the Texas licensing exams and ACM's position on licensing.

Stop That Divorce!

By Amr El-Kadi

Publications of the ACM - Viewpoint

Kadi argues that computer engineering, computer science, etc need to be kept together and proper education and licensing need to be put in place.

Licensing Software Engineers

By Dennis J. Frailey

Publications of the ACM - Viewpoint

Frailey talks about the issue of licensing and argues that Software Engineers need to stick together to solve this problem and not break down into small factions waiting to be attacked by legal groups.

A Summary of the ACM Position on Software Engineering as a Licensed Engineering Professional (Final Version)

Publications of the ACM

This document covers the findings of both task forces assigned to determine if licensing Software Engineers is a good idea

Taking the lead in Licensing Software Engineers

By Donald J. Bagert

Publications of the ACM - Viewpoint

Talks about education behind licensing Software Engineers.

Computer Science and Software Engineering: Filing for Divorce?

By Peter J. Denning

Publications of the ACM - Inside Risks

Denning talks about the difference between Software Engineers and computer scientists.

Getting your P.E. License

Reginaldo Alonzo Montague

<<http://www.allbusiness.com/human-resources/employee-development/900934-1.html>>

Covers the requirements of obtaining a P.E. license and states that a component of the exam is a general engineering component.

Why License Software Engineers?

By Dave Dorchester

IEEE Software March/April 1999

“[P]rogress is being made largely because the Texas Board decision added urgency to a long-standing problem in the software community.”

Licensing of Software Engineers in Texas

by Ken Adams

AITP Newsletter July/Aug 2003 Article 16

Argues against the licensing exam taking ACM's side while also providing some useful information about the components of the licensing exam (something ACM's papers don't really ever do).

What Do You Mean I Can't Call Myself a Software Engineer?

by John R. Speed

IEEE Software November/December 1999

This article explains the legal use of the term “engineer”, what Texas is doing, and there are mini-excerpts of what other countries are doing.

The ARIANE 5 Software Failure

by Mark Dowson

ACM SIGSOFT

Dawson talks about the ARIANE 5 failure and how it was reported as a software failure but it is actually a process failure.

Is It Time to License Software Engineers?

by Danielle Boyin

National Society of Professional Engineers (NPSE) PE Magazine, December 2007