Safe and Secure Software Systems and the Role Professional Licensure

Phil Laplante, CSDP, PE, PhD
Professor of Software Engineering
Penn State

Chair, Software PE Exam Development Committee

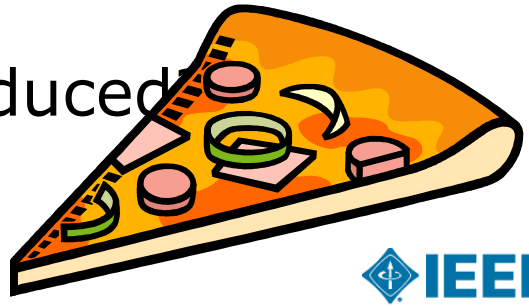
IEEE
Advancing Technology for Humanity

Outline of talk

- Identifying critical systems
- Why do we need licensure of software engineers?
- Status of US licensure project
- Challenges and unanswered questions
- Future work

A scenario

- Hot pizza vending machine explodes due to a software error – two persons are badly burned
- Original code written in basement by young entrepreneur with no formal education
- Prototype and code acquired by Big Al's Pizza Vending, Inc.
- Defect was introduced by original developer
- Who is at fault/liable?
- Could risk to public have been reduced?



Which systems affect the health, safety and welfare of the public?

- Typical domains
  - medicine, transportation, infrastructure, commerce, finance
- Typical applications
  - implantable medical devices, automobiles, elevators, power systems, financial and health record management systems
- Less-obvious
  - entertainment – e.g. amusement park ride
  - consumer goods – e.g. microwave oven
  - … etc.

Examples

Critical?	Non critical (?)
Drone aircraft	Remote controlled model airplane
Hot pizza vending machine	Soda vending machine
Robot surgery	Automated tattoo machine
Medical records system	Medical appointment self-registration system
Pension management system	Online stock trading system
Nuclear power plant	Wind power generator

Identifying questions

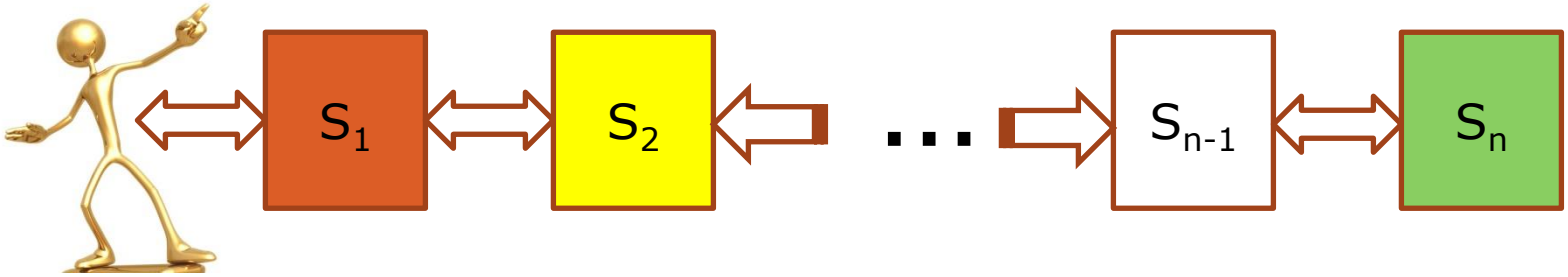
1. Does the software control a device or devices that could directly inflict harm to a human being if there was a malfunction?
2. Does the software put the assets of an individual or corporate entity at risk beyond the normal amount of risk assumed in everyday business transactions?
3. Does the software expose identifying information of an individual or a corporate entity that would violate any federal, state or local laws?
4. Does the software interact with other systems in way that directly satisfies 1-3 above?

Using the questions

- Consider: insulin pump, automotive braking, roller coaster, telemetry monitor, and water treatment plant
  - all would answer ‘yes’ to question 1.
- Consider: financial systems (e.g. tax return preparation software, e-commerce site, pension fund management system)
  - would likely answer ‘yes’ to question 2.
- What about the tax preparation software and pension fund management, e-commerce systems ?
  - might also answer ‘yes’ to question 3

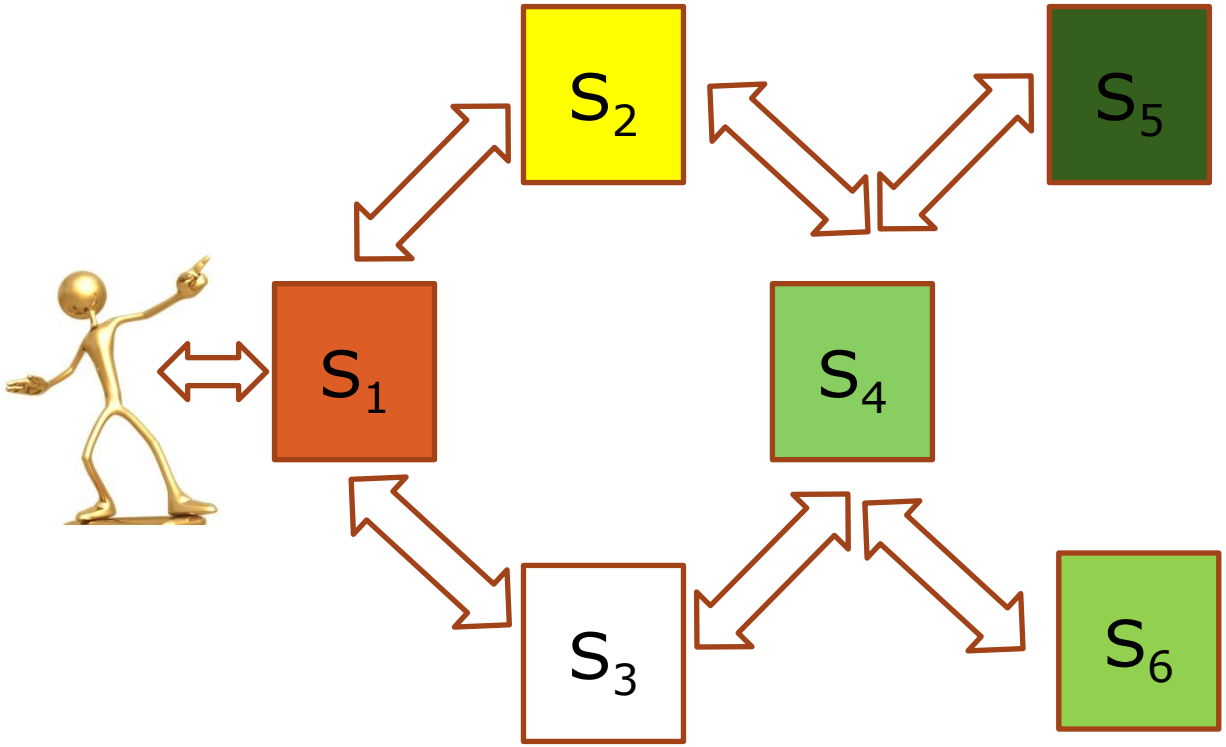
Simple interactions

- What about question number 4 – interactions?
  - a “harmless” piece of software may eventually cause a catastrophic failure



- How does failure in S_n affect S_1?
- Who is responsible?

Complex interactions



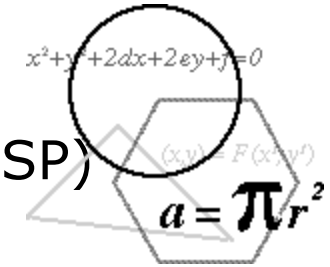
- How does failure in S_n affect S_1?
- Can security vulnerability in S_n affect S_1
- Who is responsible?

Chain of interactions

- Do we need to consider all software and the interactions – "transitive closure of safety/security"
  - e.g., a security breach to a "non-critical" system linked to a critical one causes a public disaster
  - should it be concluded that the 'non-critical" system was really "critical"?
  - What responsibility does the engineer of S_n have?
  - May have to be decided by juries and judges

Is formal modeling the answer?

- Need more sophisticated mathematical model of systems interactions e.g.
  - Church's Lambda Calculus
  - Category theory
  - Communicating Sequential Processes (CSP)
  - Milner's theory of interactions
  - Classical reliability theory



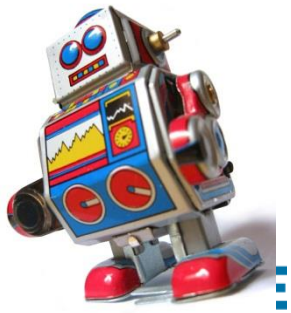
- A pure mathematical formulation legally insufficient
  - Need to consider technical, legislative, sociological, psychological, environmental, etc. factors
- More technical, legal and incident analysis needed

Third party components

- Treatment of software components produced in
  - other countries
  - open source communities
  - states where licensure is not required
  - entities that are not transparent (e.g. classified organizations)?
- Answer: same as other engineering disciplines
- e.g. licensed civil engineers spec steel produced in another country
- Same in other professions
  - Nurses
  - Accountants
  - Physicians

Asimov's Laws

- R1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- R2. A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.
- R3. A robot must protect its own existence as long as such protection does not conflict w the First or Second Law.



A possible framework for software security/safety

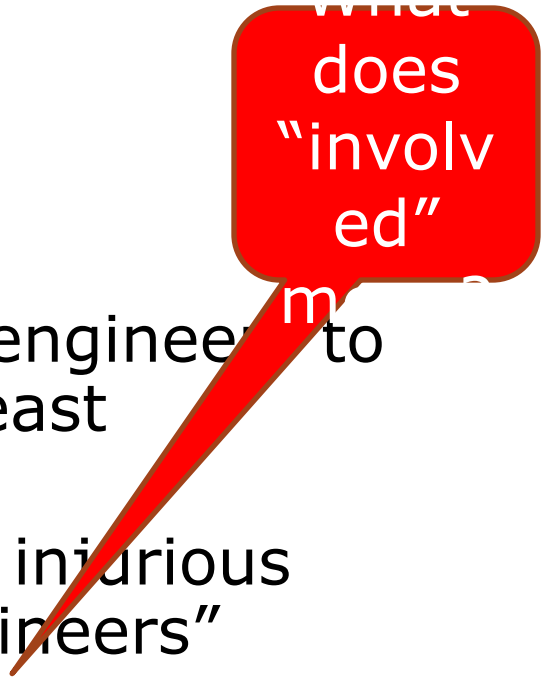
- S1. Software may not injure* a human being or, through inaction, allow a human being to come to harm
- S2. Software must respond to commands given to it by human beings, except where such inputs would conflict with S1.
- S3. A software system must protect its own existence as long as such protection does not conflict with the S1 or S2.

*“cause significant harm to health, safety, welfare or violation of privacy”

Licensure

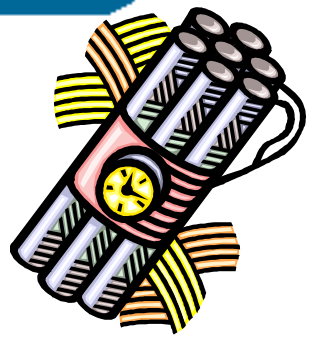
- *"The goal of a software engineer is to retire without having caused any major catastrophe." —Dilbert*
- Licensure demonstrates "minimum competency" in a discipline
- States license doctors, nurses, accountants, lawyers, engineers (…barbers, plumbers, tattoo artists, etc.)
- Certification (e.g. CSDP, CISSP, PMP) is voluntary, licensing is mandatory

Why licensure?



- States require licensure of certain engineering to ensure that any practitioner is at least minimally competent
- Intent is to protect the public from injurious consequences of incompetent “engineers”
- Licensure is required if the engineer is involved in building a system
  - whose failure could cause significant harm
  - is offering his services directly to the public
  - and not through a corporation, or government entity

Security and vulnerability



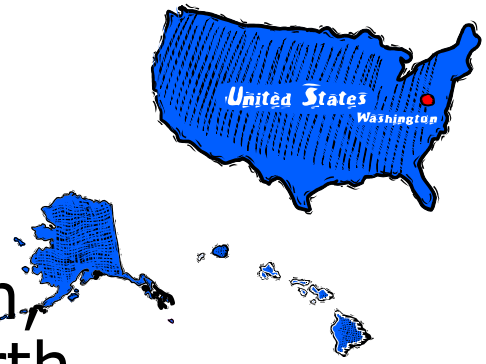
- Increased connectivity through handheld devices, smart homes, smart cars, wireless enabled devices
  - increases attack surface
- Apps and plug-ins available to the public
- Vulnerabilities inadvertently created or deliberately planted
- Security must be built into
  - quality processes
  - education
  - certification
  - licensure

The path to licensure

- Appropriate degree from an ABET-accredited program
- Fundamentals of Engineering examination
- Four years +/- of relevant experience
- Principles and Practice (PE) exam
  - This exam was the only missing item in the path to licensure for software engineers.
- Differences by states?....usually in qualification to sit
  - Years of experience
  - Waiver process, grandfathering, recognition of certifications
- Model law written and available

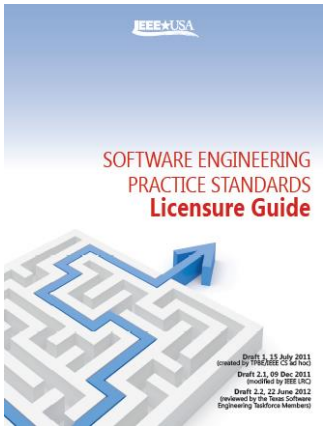
Current status of licensure

- Licensure hotly debated for years
- 1998 Texas began licensing software engineers through portfolio review
- Alabama, Delaware, Florida, Michigan, Missouri, New Mexico, New York, North Carolina, Texas and Virginia expressed interest in developing a Principles & Practices exam
- All other states and U.S. jurisdictions (District of Columbia, Puerto Rico, Guam) can offer exam
- First exam given April 2013



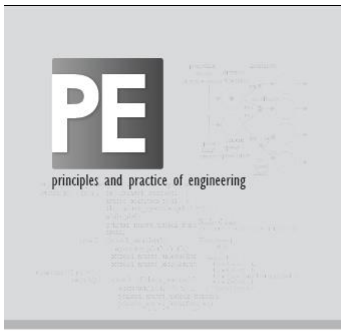
States Offering SW PE Exam in 2013

- Alabama
- Arkansas
- Colorado
- Florida
- Georgia
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Michigan
- Minnesota
- Mississippi
- Missouri
- Nebraska
- New Hampshire
- New Mexico
- North Carolina
- North Dakota
- Oklahoma
- South Carolina
- South Dakota
- Tennessee
- Texas
- Utah
- Vermont
- West Virginia
- Wyoming



P&P Exam

- Sample exam book available through IEEE
- 21 People registered for exam for 4/12/13
- Also offered in
  - Canada, UAE, Egypt, Japan, South Korea, Saudi Arabia, and Turkey.



International Perspective

- Canada, UK, Ireland, Australia, New Zealand have some kind of licensure (or chartered engineer).
- Building blocks for international recognition of licensure (e.g. Bologna Accord, Washington Accord)
- Issues of cross border practice, forum for dispute resolution, etc.
- Is software engineering practiced differently around the world?

Organizations involved in licensure effort

- NCEES
- NSPE
- IEEE – USA
- IEEE Computer Society
- Texas Board of Professional Engineers
- Prometric

Test specification: knowledge areas

	% of exam	# of items
1. Requirements	17.50	14
2. Design	13.75	11
3. Construction	11.25	9
4. Testing	12.50	10
5. Maintenance	7.50	6
6. Configuration Management	7.50	6
7. Engineering Processes	7.50	6
8. Quality Assurance	7.50	6
9. Safety, Security, and Privacy	15.00	12

Who would need a license?

- Would all software engineers need to be licensed?
  - No, only those providing their services directly to the public
- Would all software have to be developed or supervised by licensed software engineers?
  - no, only software that has an impact on the lives, property, economy, or security of people
- Licensing software engineers isn't a once-in-a-lifetime event
  - Engineers must renew their licenses annually and may be subject to mandatory continuous professional development

Source: Krutchten, 2009

How many licensed software engineers?

- Two versions of this question:
  - How many will be needed?
  - How many software professionals will become licensed?
- The first question seems harder….
- The second question…methods for estimating the eventual number of licensed professional software engineers
  - Number of software PEs in Canada – extrapolate
  - Number of CSDPs in US – extrapolate
  - Number of licensed SW engineers in Texas – extrapolate

First question: projected growth in software engineers in the US

Occupational Title	Employment, 2008	Projected Employment, 2018	Change, 2008-18	
			Number	Percent
Computer software engineers and computer programmers	1,336,300	1,619,300	283,000	21
Computer programmers	426,700	414,400	-12,300	-3
Computer software engineers	909,600	1,204,800	295,200	32
Computer software engineers, applications	514,800	689,900	175,100	34
Computer software engineers, systems software	394,800	515,000	120,200	30

Source: US Bureau of Labor Statistics

Summary

- Should licensure be required?
  - Are you willing to take personal risk on a software engineering decision?
- PEs stake their reputations, treasure, livelihood, and freedom
  - Risk tends to raise the standards of decision making
- Licensing does not prevent failures
  - licensed doctors kill patients through malpractice
  - licensed software engineers will introduce defects into software that can harm the public
- Licensure raises the standard of practice
  - provide assurance to the public of minimal competency
  - leads to safer, more secure, and more reliable software systems
- Need to better understand how to allocate responsibility and risk

Summary – continued

- Software, safety and reliability engineers and lawyers need to conduct further research leading to
  - a comprehensive system for identification of "licensable systems, that is, systems under which licensure laws apply
  - a technical and legal framework for modeling systems interactions for the purposes of fairly assigning responsibility for failure
  - a strategy for safely using third party furnished components

Relevant Publications

- Phillip A. Laplante, Beth Kalinowski, Mitchell Thornton, "A Principles and Practices Exam Specification to Support Software Engineering Licensure in the United States of America," *Software Quality Professional*, vol. 15, no. 1, January, 2013, pp. 4-15.
- Phillip A. Laplante, "Safe and Secure Software Systems: The Role of Professional Licensure, "*IT Professional*, vol. 14, no. 6, 2012, pp. 51-53.
- P. A. Laplante, "An International Perspective on U.S. Licensure of Software Engineers, "Technology *and Society*, vol.32, no.1, pp.28-30, Spring 2013

Questions?

Contact: plaplante@psu.edu