ECE450 – Software Engineering II
-new** and **improved-

Today: Why Software **Engineering**?

Reminder

- Form teams of 3-4 people
- Fill in the Team Formation sheet
- Return by Monday at the latest
- Assignment 1 is out

Why Software **Engineering**?

- You're engineers...
- Frankly, aren't you offended?
- Can you approach building software as building a bridge? Why? Why not?
- How is software engineering like other engineering disciplines?

Software **Engineering**: Origins of the term

- Back in the 60's, hardware was much more expensive than software
- 1968 NATO *Software Engineering* Conference
  - To imply "the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering"
  - Term caught on instantaneously
    - We've been using it since

Interpretations of the term

- Wishful thinking
  - "I wish software development was as structured and professional as the engineering disciplines"
  - Translation: We have a mess over here. Our neighbours seem to have their place all tidied-up. Let's become like them!
  - Risk: "Cargo cult" engineering

Interpretations of the term (cont)

- Application of the "parent" science
  - For other engineers, the "parent" may be Physics, Chemistry, etc.
  - For software, the "parent" is Mathematics
    - Formal Methods
  - Translation: Just as mankind reached applicability of science through engineering, we should reach the applicability of math and computer science through *software* engineering
  - Risk: Assumes "parent" science is math!

Interpretations of the term

- Metaphor
  - Treat requirements and developers as you would treat the components of a manufacturing line
  - Metaphorist utopia: Developers are interchangeable pieces, development process is a production line
    - Measure >> Improve >> Measure >> ...
  - Risk: Bad metrics; creative minds don't respond well when treated like cogs.

"Engineering" = "we're serious"?

- Confusion over the term as academics and industry use it to refer to any serious approach to a soft field:
  - Software Engineering
  - Requirements Engineering
  - Knowledge Engineering
  - Usability Engineering
  - Information Engineering
  - Cognitive Engineering
  - ...!?!?!

The many worlds of Software Engineering

- When was the last time a chemical engineer designed a building?
  - Specialization
- Software engineers often switch domains
  - Enterprise systems
  - Web development
  - Videogames
  - Custom-made database applications
  - Manufacturing software
  - ...

Is there engineering in SE?

- IEEE definition
  - Software engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software.
- Necessary?
- Desirable?
- Possible?

Software Engineering – **Yea!**

- Do you want to put your life in the hands of informal, chaotic processes?
  - Would you ride an airplane if you knew its spec was described in a napkin?
- A person kills retail, we can kill wholesale!
- We depend on correct software:
  - Airplanes
  - Medical software
  - Dams, nuclear plants
  - Military equipment

Software Engineering – **Yea!** (cont)

- Less dramatically
  - Bad software leads to miscommunications, money loss, frustration, productivity loss, and a long etcetera.
- You can't control what you can't measure
- Not only desirable, but *possible*
  - Programs are mathematical entities
    - We can prove them right or wrong
  - Projects are akin to manufacturing
    - State requirements carefully and in detail, and any developer can code them and deliver the system

Software Engineering – **Nay!**

- Do you want to put your life in the hands of people who believe they can *prove* the correctness of complex software systems?
  - Your record is missing, which means you don't exist
  - Cold War missile radar incident
- Keep humans "in the loop"!
- We do not need "correct" software
  - Google
  - Microsoft
  - Your favourite start-up

Software Engineering – **Nay!** (cont)

- Rigid approaches make for unappealing products
  - Creativity sucked out of the code and developers
- You control what you can't measure, all the time
- Not only undesirable, but *impossible*
  - Programs are not mathematical entities
    - Since they need to respond to an uncertain world
  - Projects are not akin to manufacturing
    - Can we specify precisely the next killer app?

Software Engineering Yea or Nay?

- Your thoughts?

Radical vs. Normal Design

- Normal design
  - We know how the device works, when does it work, what makes it fail
    - Cars
    - Computers
    - Email clients

Radical vs. Normal Design

- Radical design
  - Innovative practices, novel requirements
    - First cars
    - First computers
    - First email clients
  - Error-prone!
- Is software mainly radical or normal design?

Doing it “more like engineering”

- “The trouble with using engineering as a reference is that we, as a community, don’t know what that means. (...) In my travels, people mostly use the word *engineering* to create a sense of guilt for not having done enough of something, without being clear what that something is. (...) When people say “Make software development more like engineering,” they often mean, “Make it more like running a plant, with statistical quality controls.”
  - Alistair Cockburn, “Agile Software Development”

Engineering => Certification?

- Can we encapsulate all required facts on software engineering in a certification test?
  - And do those facts exist?
- Will a certified software engineer be more trustworthy than an uncertified one?
  - Should they work only in a prescribed domain?
- Perhaps when we have a real software catastrophe or two the pro-certification voices will prevail?
- Your thoughts?

Liability of Software Engineers

- Should software engineers be liable for the errors in their systems?
  - Lives lost?
  - Monetary losses?
  - Data loss?
- Current software licenses free software developers of any responsibility in the use of their products

ACM/IEEE Code of Ethics

(available at http://www.acm.org/serving/se/code.htm)

- PUBLIC: Act consistently with the public interest
- CLIENT AND EMPLOYER: Act in the best interest of client and employer, consistent with the public interest
- PRODUCT: Ensure product meets highest professional standards
- JUDGMENT: Maintain integrity and independence in professional judgment
- MANAGEMENT: Managers and leaders should promote an ethical approach to the management of software development
- PROFESSION: Advance the integrity and reputation of the profession
- COLLEAGUES: Be fair and supportive of colleagues
- SELF: Participate in lifelong learning, promote ethical approach to the profession

Alternative views

- Software Craftsmanship
  - Apprenticeship tradition
  - Develop your own style
  - Find a niche and excel at it
  - Become a mentor
  - But
    - Not consistent; dependent on human variability
    - Not repeatable
    - Not formalizable

Alternative views (cont)

- Software Writing
  - But one writes individually!
- Software Painting
  - "Hacker" as an artist
  - http://www.paulgraham.com/hp.html
- Software Accretion
  - Oyster pearl growth, incremental development
- Software as Design
  - Radical design, not repetition
  - Architecture, not Civil Engineering

In this course...

- When I talk about *Software Engineering* I mean
  - A careful approach to software development issues
    - The "we're taking it seriously" stance
    - Synonym with "professional software development"
    - Really what most people in industry/academia have in mind
- I do not mean
  - That quantitative approaches are always necessary
    - Or possible
  - That software should be based on mathematics and proved correct to move on
- Your approach may vary

Software is not what we think it is

- Software != Code
  - Source code is a *subset* of the things that make a software project
    - Requirements and specification documents
    - Design documents
    - Test suites, test plans
    - Interfaces
    - Documentation
  - *"For civilian projects, at least 100 English words are produced for every source code statement in the software. Many new software professionals are surprised when they spend more time producing words than code"*
    - Capers Jones, "Gaps in Programming Education", IEEE Computer, 1995.

Large Software is not what we think it is

- Large Software != Lots of Code
  - You wish!
  - Large software brings up several issues of scale:
    - Total comprehensibility must be forfeited
    - Solo programming becomes impossible
    - Communication becomes an essential challenge
    - Changing requirements are an everyday reality
    - Lifetime is measured in years or decades
    - People don't know what they want

Ideal goals of Software Engineering

- Correctness
  - No errors
- Usefulness
  - Does what we want it to do
- Minimal production effort
  - And therefore,
    - minimal cost
    - minimal time
    - maximum profit
- Minimal maintenance effort

Software Qualities

- Usefulness
- Modularization
- Reliability
- Robustness
- Usability
- Efficiency
- Flexibility
- Good internal documentation
- Good external documentation

Software Qualities are conflicting

- First, they are all expensive
  - So they all conflict with resource optimization
- Second, they are conflicting between each other
  - Performance vs. Modularization
  - Reliability vs. Flexibility
- Can't have it all: need to make trade-offs
  - Prioritization becomes essential