Why Software Engineering?

Dr John H Robb, PMP
UT Arlington
Computer Science and Engineering

Questions to Discuss

- What is Software Engineering?
- Why is it important?
- How is software engineering different from programming?
- Who in the class has had experience with UML? What aspects of UML?
- Does anyone use UML?
- How does UML fit within our class and within Software Engineering?
- What alternatives are there to UML?

Software Engineering

- Software engineering is the systematic application of studied, refined, and time proven methods to build reliable software

- When is programming a better fit than software engineering? Possibly when developing small (individual developer) applications...

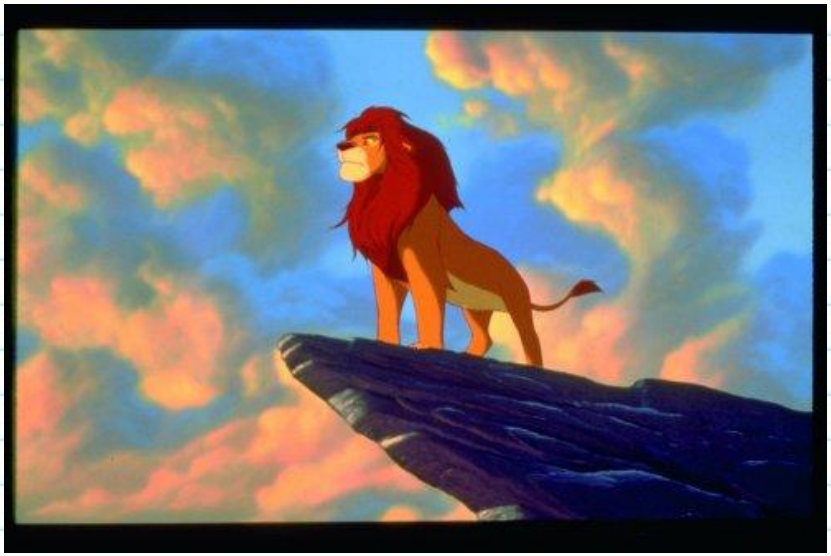
- Why can't programming solve medium to large complexity problems?
  - Too much time spent on coding instead of developing what the customer needs - no focus on defining key features and priorities
  - Its focuses is on the leaves or trees - not on the forest - the architecture is clumsy or non-existent - big picture analysis not done
  - No thought is given to problem partitioning - can't effectively use multiple people to develop the software
  - Programming is code centric, but successful software involves the whole life cycle (requirements, design, estimation/tracking, test)
  - Maintenance is the biggest part of software - use of proven design principles make it easier to maintain

People Working Together

- Software engineering is the process of many people working together and putting together many versions of large and complex software
  - In Civil Engineering medium to large construction projects are performed by teams - is software any different?
  - Software team related issues include
    - How do we partition work so that teams can work together?
    - How do we combine partitioning of software products with stepwise refinement? How do we communicate this?
    - How do we keep an entire team efficiently working to deliver successive product updates?
- Software engineering is
  - the study of developing software by teams of people - it answers the question how it can be done more efficiently, effectively, reliably, etc.
  - similar to how Industrial Engineering improves the factory or production process

Programming vs. Software Engineering

- Programming doesn't place emphasis on studying what has happened before, where the pitfalls are and how to avoid them
- Software Engineering is learning from other's failures and successes to avoid the most common pitfalls



Software Engineering tells us that the most sure approach to failure on medium to large scale applications is to start coding it first

Why UML?

- Software development is all about going from an idea in someone's head to a working system. These thoughts can require some complex solutions to make them happen.
- How do we solve complex problems?
  - We look at the overall problem and break it into smaller pieces (partitioning) - we try to determine the best way to break this up
  - We transform it from the abstract to the more concrete - we do this by considering a few things at a time so that we can get it right - we capture each of these transformation as we do them
  - We apply studied, refined, and proven techniques as we go through each of these steps
- UML is the only widely available solution for complex software problems
  - $Bs have been invested in it to make it the effective medium it is today
  - It has been defined, studied, refined, and improved over the last 15+ years - it is time proven medium

Are There Alternatives to UML?

- The Software Engineering Institute (SEI) is one of the most recognized Software Engineering organizations in the world - they study software development techniques, approaches, efficiency, problems, etc
- It's easy to say "Don't use UML," but hard to say "Use this instead" - there are no widely acclaimed substitutes for UML.
- Informal notations? They help in many contexts but in others the inherent ambiguity impairs the ability to understand the design.
- No design documentation? The lack of documentation becomes a problem when the design has to be communicated across time (to a developer who will join the team later) or space (to a developer working in another location).
- Domain-specific languages (DSLs) for visual modeling? These have not found broad adoption.
- (Formal) Architecture description languages (ADLs)? Some formal ADLs, such as AADL, have found niches only.

Why is the Use of UML So Controversial?

- SEI - the top-5 reasons people don't use UML
  1. Don’t know the notation
  2. UML is too complex (there are 14 different notations!)
  3. Never took a UML class
  4. Informal notations do just fine
  5. Architecture documentation not deemed important

- SEI - six reasons to use it
  1. It’s the standard
  2. Tool support
  3. UML is flexible
  4. UML models are portable
  5. Very rich - typical developers consistently use a subset
  6. Architecture is important

- Don't be a Cowboy coder - software has a counter culture too!

- A list of UML tools - UML is very widespread and here to stay
- https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Class On Software Engineering

- UML is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system. There is no widely used alternative!
- UML is simply the language used to **communicate** the piecewise decomposition of a software system - our focus is on Software Engineering
- In this class we study Software Engineering to help us develop a systematic approach to analyze, design, build, and support a medium to large complexity software product
  - We're studying some of the most applied, refined and proven software techniques that exist for these types of problems
  - We'll acknowledge that small problems can be done without some of these steps - but perhaps not all
  - We're going to trivialize the coding part and place more importance on getting the system right

Future Technologies Will Emphasize Large Scale Software Development

Seamless Mesh of Intelligent Devices

IEEE Computer Society 2022 Report - 23 Computing Technologies That Will Shape the World in 2022

1. **3D Printing**
2. **Big Data and Analytics**
3. Open Intellectual Property Movement
4. Massively Online Open Courses
5. **Security Cross-Cutting Issues**
6. Universal Memory
7. 3D Integrated Circuits
8. Photonics
9. **Cloud Computing**
10. Computational Biology and Bioinformatics
11. Device and Nano-technology
12. Sustainability
13. **High Performance Computing**
14. **The Internet of Things**
15. Life Sciences
16. **Machine Learning and Intelligent Systems**
17. **Natural User Interfaces**
18. **Networking and Inter-connectivity**
19. **Quantum Computing**
20. **Multi-core**
21. **Software Defined Networks**
22. **Robotics**
23. **Computer Vision & Pattern Recognition**

Legend: Software Intensive

IEEE Computer Society 2022 Report - 23 Computing Technologies That Will Shape the World in 2022

Discussion of the software involved technologies

- 3D Printing - ability to change the production line and ability to print biological items (organs)
- Big Data and Analytics - rapid changes in data acquisition, storage, and processing technologies; complex privacy issues.
- Security Cross-Cutting Issues - tradeoff decisions to be made about privacy versus security
- Cloud Computing - Need to develop a “divide-and-conquer” approach to standardizing layers of building block services
- High Performance Computing - enable existing applications to increase parallelism
- The Internet of Things - reach from anywhere to anywhere
- Machine Learning and Intelligent Systems - Intelligent assistants everywhere (e.g. refrigerators) - autonomous cars
- Natural User Interfaces - touch, gesture, and speech need to be integrated to provide a more natural and efficient way of interacting

IEEE Computer Society 2022 Report - 23 Computing Technologies That Will Shape the World in 2022 (cont.)

- Networking and Inter-connectivity - the ability for any device to connect to anything anywhere
- Quantum Computing - integration of QC within large classic computing infrastructures
- Multi-core - wearable IT systems, smart-phones, cameras, games, automobiles, medical systems such as drinkable inner-cameras for health diagnosis,
- Software Defined Networks - the developer of a distributed system will develop a virtual network as an integral part of his system, where it is part of his application
- Medical Robotics - robot assisted surgery, treatment or diagnosis
- Computer Vision & Pattern Recognition - autonomous actors—vehicles, household care robots, search and rescue robots, and so on—need to know what is going on around them

Key Point - Software Engineering Is Very Important and Will Play An Even Larger Role In The Future

Recent Worldwide Software Engineering Initiatives

- World-wide initiatives have emerged to promote Software Engineering as a specific body of knowledge with key areas of focus
  - Software intensive technologies continue to grow and directly interact with people, both safety and security concerns arise
  - Software failures have been highly visible and in some cases quite catastrophic https://en.wikipedia.org/wiki/List_of_software_bugs
  - These initiative are intended to address both safety and security concerns and provide more reliable software
- Both the ACM and IEEE have developed specific areas of focus for Software Engineering
  - IEEE Software Engineering Body of Knowledge (SWEBOK) - used to obtain Software Engineering Certifications
  - ACM has developed the Curriculum Guidelines for Graduate Degree Programs in Software Engineering (GSwE2009)

Recent Worldwide Software Engineering Initiatives (cont.)

- Where Are These Initiatives Headed?
  - Originally, the ACM withdrew its support of the IEEE SWEBOK stating that it could not support licensure of software engineers
  - There is considerable controversy over licensure, but many influential leaders are now recanting their criticism due to the growing nature of software intensive systems (e.g., Plante, Cerf)
  - In the US, 40 of 50 states are in the process or have already created a path for the licensure of software engineers - the State of Texas uses the IEEE SWEBOK Certification test
  - Most likely there will not a requirement to obtain a license, but understanding the body of knowledge and having a license will be very beneficial for software engineers to have
  - This really underscores the importance of software engineering curriculum and this class to your career

Software Engineering Already Plays A Key Role In Society And Will Continue To

SWEBOK V3 - 2014

- The Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) has been created through cooperation among several professional bodies and members of industry and is published by the IEEE Computer Society (IEEE)
- The standard can be accessed freely from the IEEE Computer Society.
- The Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) describes generally accepted knowledge about software engineering.
- Its 15 knowledge areas (KAs) summarize basic concepts and include a reference list pointing to more detailed information.
- For SWEBOK Guide V3, SWEBOK editors received and replied to comments from approximately 150 reviewers in 33 countries.

SWEBOK V3 - 2014 (cont.)

- The published version of SWEBOK V3 has the following 15 knowledge areas (KAs) within the field of software engineering:
  1. Software Requirements
  2. Software Design
  3. Software Construction
  4. Software Testing
  5. Software Maintenance
  6. Software Configuration Management
  7. Software Engineering Management
  8. Software Engineering Process
  9. Software Engineering Models And Methods
  10. Software Quality
  11. Software Engineering Professional Practice
  12. Software Engineering Economics
  13. Computing Foundations
  14. Mathematical Foundations
  15. Engineering Foundations

Curriculum Guidelines for Graduate Degree Programs in Software Engineering

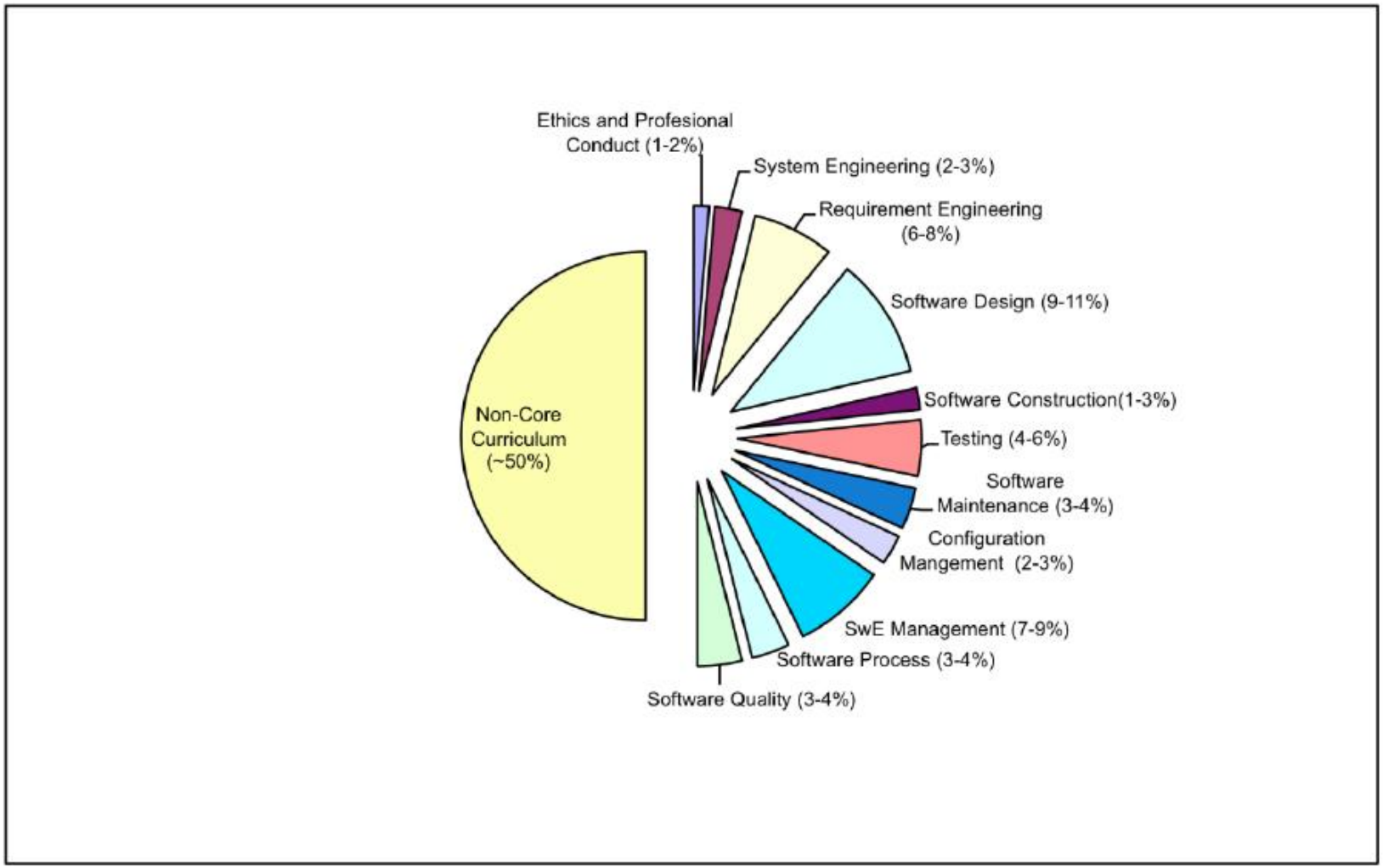
- Curriculum Guidelines for Graduate Degree Programs in Software Engineering - GSwE2009 (published by the ACM) was created to:

- Improve existing graduate programs in SwE from the viewpoint of universities, students, graduates, software builders, and software buyers;

- Enable the formation of new graduate programs in SwE by providing guidelines on curriculum content and advice on how to implement those guidelines; and

- Support increased enrollment in graduate SwE programs by increasing the value of those programs to potential students and employers.

Curriculum Guidelines for Graduate Degree Programs in Software Engineering



Correlation Across Stds

IEEE SWEBOK V3	ACM GSwE2009 CBOK
Software Maintenance	Software Maintenance
Software Engineering Process	Software Engineering Process
Software Testing	Testing
Software Quality	Software Quality
Software Design	Software Design
Software Engineering Management	Software Engineering Management
Configuration Management	Configuration Management
Software Construction	Software Construction
Software Requirements	Requirements Engineering
Software Engineering Professional Practices	Ethics and Professional Conduct
Computing Foundations	Computing Fundamentals
Mathematical Foundations	Mathematical Fundamentals
Engineering Foundations	
Software Engineering Models and Methods	
Software Engineering Economics	
	Systems Engineering
	Software Engineering

What We Cover in This Course

Total list of SWE skills	CSE 5324 covers
Software Engineering Process	Yes
Software Engineering Models and Methods	Yes
Software Requirements	Yes
Systems Engineering	Yes
Software Engineering	Yes
Software Design	Yes
Software Construction	Some
Software Maintenance	Yes
Software Quality	Yes
Software Testing	Yes
Software Engineering Management	Some
Software Engineering Economics	Some
Configuration Management	Some
Software Engineering Professional Practices	No
Computing Foundations	No
Mathematical Foundations	No
Engineering Foundations	No

What We Cover in This Course (cont.)

- One of the primary criticisms of the SWEBOK V3 is that it does not adequately address the Agile process
  - Agile is the most prevalent process used today in software development
  - SWEBOK V3 briefly mentions this but provides little detail
- Due to the prominence of Agile, this class utilizes an Iterative approach very similar to the Agile process
  - The class has 3-one month long iterations much like many agile processes
  - We have specific lectures dedicated to the Agile process and each lecture module reviews the process discussed and how it applies to Agile
  - We also utilize some Agile planning techniques in planning each iteration's content
- This should provide you with some hands-on experience with Agile when developing the class project

Textbook Related Slides

Software Development

- A software development process transforms the initial system concept into the operational system running in the target environment.
- It identifies the business needs, conducts a feasibility study, and formulates the requirements or capabilities that the system must deliver.
- It also designs, implements, tests, and deploys the system to the target environment.
- The biggest question we will address this semester is "how do I develop a software product from a customer concept into a working product?"

Software Quality Assurance

Software quality assurance (SQA) ensures that

- the development activities are performed properly, and
- the software artifacts produced by the development activities meet the software requirements and desired quality standards.

Software Project Management

- Software project management oversees the control and administration of the development and SQA activities.
- Project management activities include
  - effort estimation
  - project planning and scheduling
  - risk management
  - project administration, and
  - others.

These activities ensure that the software system is delivered on time and within budget.

Object-Oriented Software Engineering

- Object-oriented software engineering (OOSE) is a specialization of software engineering.
- The object-oriented paradigm views the world and systems as consisting of objects that relate and interact with each other – my takeaways on OOSE follow
  - It provides the ability to relate the software system being developed back to the customer in their terms
  - OOSE allows the customer to conceptualize users, roles, classes (objects), and top level interactions from their point of view – it eliminates a huge conceptual gap
  - It builds a conceptual model for the developer of how the customer's world operates
- OOSE encompasses:
  - OO processes
  - OO methodologies
  - OO modeling languages
  - OO tools

Key Takeaway Points

- Software engineering aims to significantly improve software productivity and software quality while reducing software costs and time to market.

- Software engineering consists of three tracks of interacting life cycle activities:
  - software development process
  - software quality assurance, and
  - software project management

- Object-oriented (OO) software engineering is a specialization of software engineering. It helps to effectively bridge the communication gap between developer(s) and customer(s)

A Word About Your Project

- Java JSP/Android development is not a formal part of the class objectives
  - Software Engineering techniques help with Android development just as well as developing desktop, web or enterprise software applications
  - This turns out to provide a great mechanism for working with your teammates to learn Android development together
  - We provide you with some lectures and other materials to help in this but graduate classes have an expectation of more focused self-study
  - You will develop a working app and formally demonstrate it at the end of the semester - it must be a working product
  - The executable project does not get formally graded - what is formally graded are the class techniques used to develop your project

Teamwork

- Team projects are critical to software engineering because almost all industry work is done in teams - e.g., Agile “Scrum”
- No other part of the class can pose as much difficulty as a non-functional team
- Signs of a non-functional team
  1. Non-responsive to team emails about assignments and meetings
  2. Students miss team meetings - with or without excuses
  3. Students do not contribute toward the teams deliverable project
  4. Checklists are not used for each deliverable
  5. Team members turn in assignments late
  6. Team members do not review each others work
  7. No one is performing an overall role to integrate the product into a deliverable product
- Teamwork will decide your class grade - your grade will be no higher than the effort your team produces

Teamwork (cont.)

- Signs of a great functional team
  1. A plan is developed for the project deliverable in work - this plan reserves at least 48 hours for final review and corrective updates
  2. Team members respond to team emails about assignments
  3. Team members attend all meetings
  4. All team members contribute toward the teams deliverable project
  5. Team members use checklists before submitting the project to their peer reviewer
  6. Peer reviewer reviews team members work according to the checklist with constructive feedback
  7. Team member being reviewed accepts the constructive feedback
  8. Someone is performing an overall role to integrate the product into a deliverable product

Teamwork (cont.)

- I want a pledge from each person in the class that you will work as hard as you can toward your team project - no slackers!
- **Please drop the class if you do not intend to work hard for your team**
- I want you to appoint a team Captain for each team, this person will
  1. Coordinate with team members on team assignments, team meetings
  2. The team Captain needs to politely but forcefully “raise the bar” - to expect and receive each team members best inputs
  3. The team Captain is required to send an email to the Professor about non-participants
- Non-participants will get docked points from their team grade
- Any team member harassing the team Captain for doing their job will also have points deducted from their grade
- I want you to learn and work well in your teams