605.601 Foundations of Software Engineering
Fall 2020

Module 01: An Overview

Dr. Tushar K. Hazra

tkhazra@gmail.com

(443)540-2230

605.601 Foundations of Software Engineering

Topics for Discussion

- Why study Software Engineering?
- Software and Software Engineering: Definitions
- Software vs. Hardware: What costs more?
- Software Engineering as a Profession – A Discipline
- Software Engineering for Us

Why Study Software Engineering?

- Industry growth and demand for software engineers
- Highly sought after profession and a discipline
- Abstracts and encompasses the knowledge and applies the concepts of computer science
- Large software systems are difficult to create but not due to any one problem
  - Challenge is to find real solutions to real problems on actual schedule with available resources

Image source:
http://commons.wikimedia.org/wiki/File:La_Brea_Tar_Pits.jpg

605.601 Foundations of Software Engineering

- Ariane 5
  - Flight 501 video: https://youtu.be/gp_D8r-2hwk
    - Reused guidance system from Ariane 4
    - Limited testing due to prior success
  - Integer overflow in computation threw unhandled exception
    - Primary and backup inertial reference systems halted
    - Diagnostic error data interpreted as incorrect position and velocity
    - Rockets attempted to correct for deviation that had not occurred

Image source: https://commons.wikimedia.org/wiki/File:Ariane_5ES_with_ATV_4_on_its_way_to_ELA-3.jpg

605.601 Foundations of Software Engineering

- Ariane 5 Flight 501 Disaster
- What software engineering failures led to the disaster?
  - **Project Management** Inadequate review processes without "clear-cut authority and responsibility"
  - **Requirements** Ariane 4 alignment code operational but not traceable to any requirement for Ariane 5
  - **Design** Specification allowed for hardware but not software failures
  - **Implementation** Overflow when converting 64-bit floating point number to a 16-bit signed integer
  - **Testing** No test for the flight time and trajectory of Ariane 5
  - **Risk Management** Original decisions for Ariane 4 were not revisited when requirements changed for Ariane 5

Source: Nuseibeh, B. (1997). Ariane 5: Who Dunnit? IEEE Software, 14(3):15–16

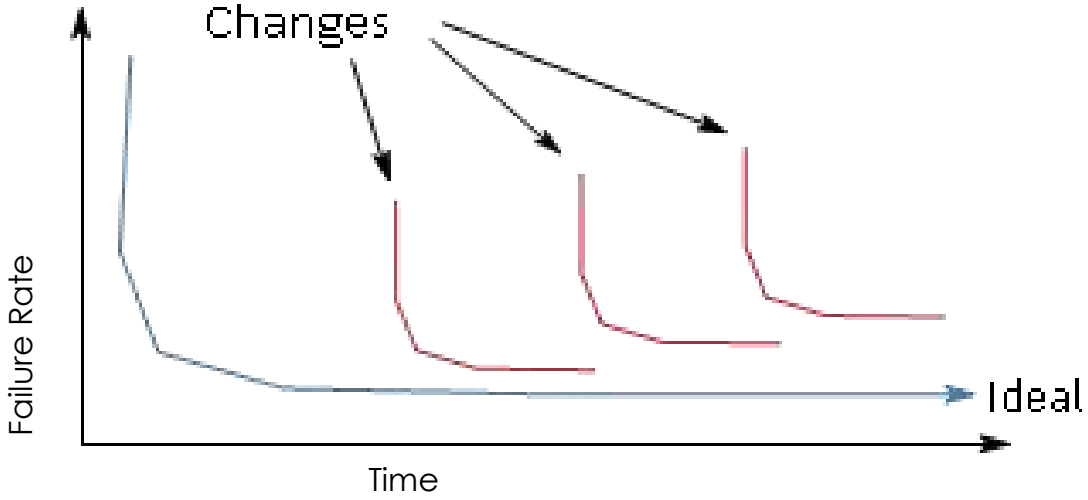
What is Software?

Definition of Software (from Sommerville):

- Computer programs and associated documentation such as requirements, design models and user manuals
- Software products may be developed for a particular customer or may be developed for a general market
- Software products may be
  - Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word
  - Custom - developed for a single customer according to their specification
- New software can be created by developing new programs, configuring generic software systems or reusing existing software

605.601 Foundations of Software Engineering

- Software
  - Developed or engineered, it is not manufactured in the classical sense
  - Doesn't "wear out"
  - Although the industry is moving toward component-based construction, most software continues to be custom-built



605.601 Foundations of Software Engineering

- Legacy Software
  - Why must it change?

  Software must be
    - adapted to meet the needs of new computing environments or technology
    - enhanced to implement new business requirements
    - extended to make it interoperable with other more modern systems or databases
    - re-architected to make it viable within a network environment

605.601 Foundations of Software Engineering

- Joys of Programming
  - Creativity; the joy of making things
  - Helpfulness, usefulness to others
  - Ability to fashion complex components together to work in tandem
  - Constantly learning, always something new (practical or theoretical)
  - Inherently tractable medium; software is flexible and changeable

605.601 Foundations of Software Engineering

- Woes of Programming
  - Software must perform perfectly (i.e., computer does exactly what it's told)
  - Others determine objectives, provide resources, and furnish information
    - Dependencies are often poorly designed, implemented, and documented
  - Debugging is tedious, painstaking labor
  - Software projects converge more slowly as the end approaches
    - Expectation that convergence will be faster
    - Constant belief that the last bug has been found
  - Products are quickly obsolete

605.601 Foundations of Software Engineering

- Software Products [Brooks, 1995]
  - Program Complete unit of software that solves a particular problem
  - Programming Product Generalized *program* with thorough testing and documentation; maintainable by others
  - Programming System *Program* with well-defined interfaces that facilitate reuse in a larger system; thorough integration testing
  - Programming System Product Most useful software product that comprises *programming product and programming system*

605.601 Foundations of Software Engineering

- The vast majority of software today is handcrafted by artisans using craft-based techniques that cannot produce consistent results. These techniques have little in common with the rigorous, theory-based processes characteristic of other engineering disciplines. As a result, software failure is a common occurrence, often with substantial societal and economic consequences. Many software projects simply collapse under the weight of unmastered complexity and never result in usable systems at all. [Prowell et al., 1999]

Definition of Software Engineering

Software Engineering

- An engineering discipline that is concerned with all aspects of software production
- Software engineers should adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

Additional Definitions of Software Engineering

- Definitions of Software Engineering
  - Definition (Software Engineering)
    [Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines. [Bauer, 1972, emphasis added]
- 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.
  - The study of approaches as in 1. [IEEE, 1990, emphasis added]

Software Engineering

- Software engineering provides a process for. . .
  - Engineering
    - Communication
    - Planning
    - Modeling
    - Analysis of requirements
    - Design
  - Construction
    - Code generation
    - Testing
  - Deployment
- Management
  - Software project management
  - Formal technical reviews
  - Quality assurance
  - Configuration management
  - Work product preparation and production
  - Reusability
  - Measurement
  - Risk management

Software Engineering

- Realities
  - A concerted effort should be made to understand the problem before a software solution is developed
  - Design is a pivotal activity
  - Software should exhibit high quality
  - Software should be maintainable

Figure: Unplanned vs. disciplined approaches to software

Image source: http://commons.wikimedia.org/wiki/File:Spaghetti_all'_arrabbiata.jpg
Image source: http://commons.wikimedia.org/wiki/File:Mexican_lasagne_with_parsley,_April_2011.jpg

605.601 Foundations of Software Engineering

- How It all starts
  - Every software project is precipitated by some business need:
    - the need to correct a defect in an existing application;
    - the need to adapt a "legacy system" to a changing business environment;
    - the need to extend the functions and features of an existing application; or
    - the need to create a new product, service, or system

How it All Starts…

- Software Development Process
  - What do we mean by Software Development Process?
  - What is software development life cycle?
- Current and emerging trends

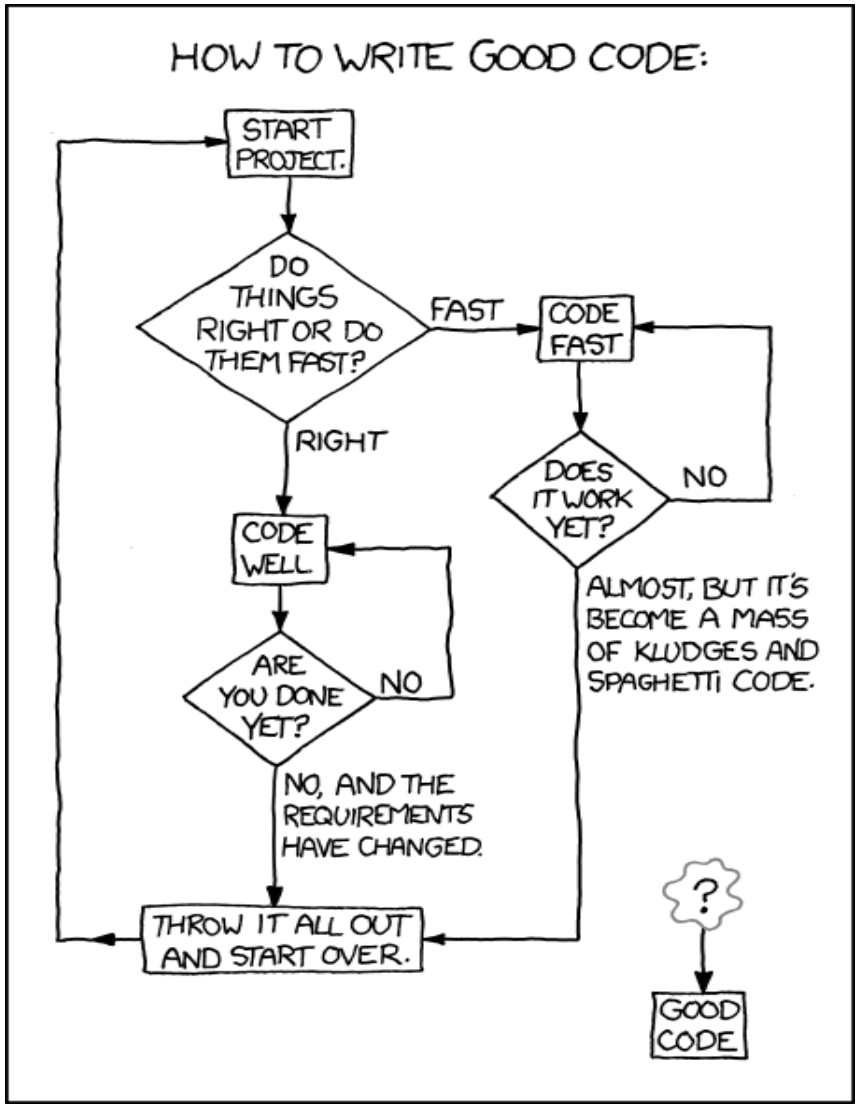
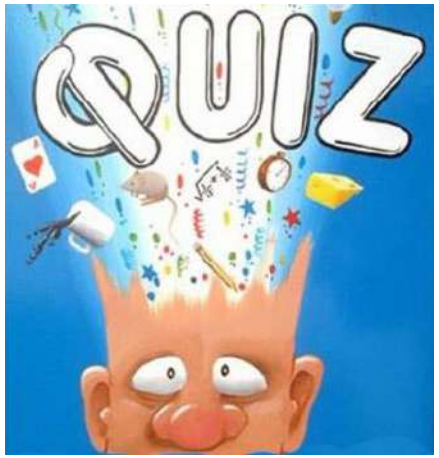


Image source: https://xkcd.com/844/

What Costs More? Software or Hardware?



Work with your classmate to determine what costs more for your laptop

- Software or hardware?
- And, how much more?

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs
- Software engineering is concerned with cost- effective software development

Software Engineering as a Profession – A Discipline

- It is different from Computer Science
  - Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software
  - Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering)

Software Engineering as a Profession – A Discipline

- It is different from System Engineering
- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system
- System engineers are involved in system specification, architectural design, integration and deployment

Software Engineering as a Profession – A Discipline

Professional and ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals
- Ethical behavior is more than simply upholding the law

Software Engineering as a Profession – A Discipline

Software Engineering Code of Ethics

- The professional societies such as ACM and IEEE (in the US) have cooperated to produce a code of ethical practice
- Members of these organizations sign up to the code of practice when they join
- The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession

Intellectual Property (IP) rights and Computer misuse or two of the major issues…

Software Engineering for Us

- The Essence of SE Practice As Polya [Polya, 1945] suggests

1. Understand the problem ⇒ communication and analysis
2. Plan a solution ⇒ modeling and software design
3. Carry out the plan ⇒ code generation
4. Examine the result for accuracy ⇒ testing and quality assurance

Software Engineering for Us

- Understand the Problem
  - Who has a stake in the solution to the problem? That is, who are the stakeholders?
  - What are the unknowns? What data, functions, and features are required to properly solve the problem?
  - Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand?
  - Can the problem be represented graphically? Can an analysis model be created?

Software Engineering for Us

- Plan the Solution
  - Have you seen similar problems before? Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
  - Has a similar problem been solved? If so, are elements of the solution reusable?
  - Can subproblems be defined? If so, are solutions readily apparent for the subproblems?
  - Can you represent a solution in a manner that leads to effective implementation? Can a design model be created?

Software Engineering for Us

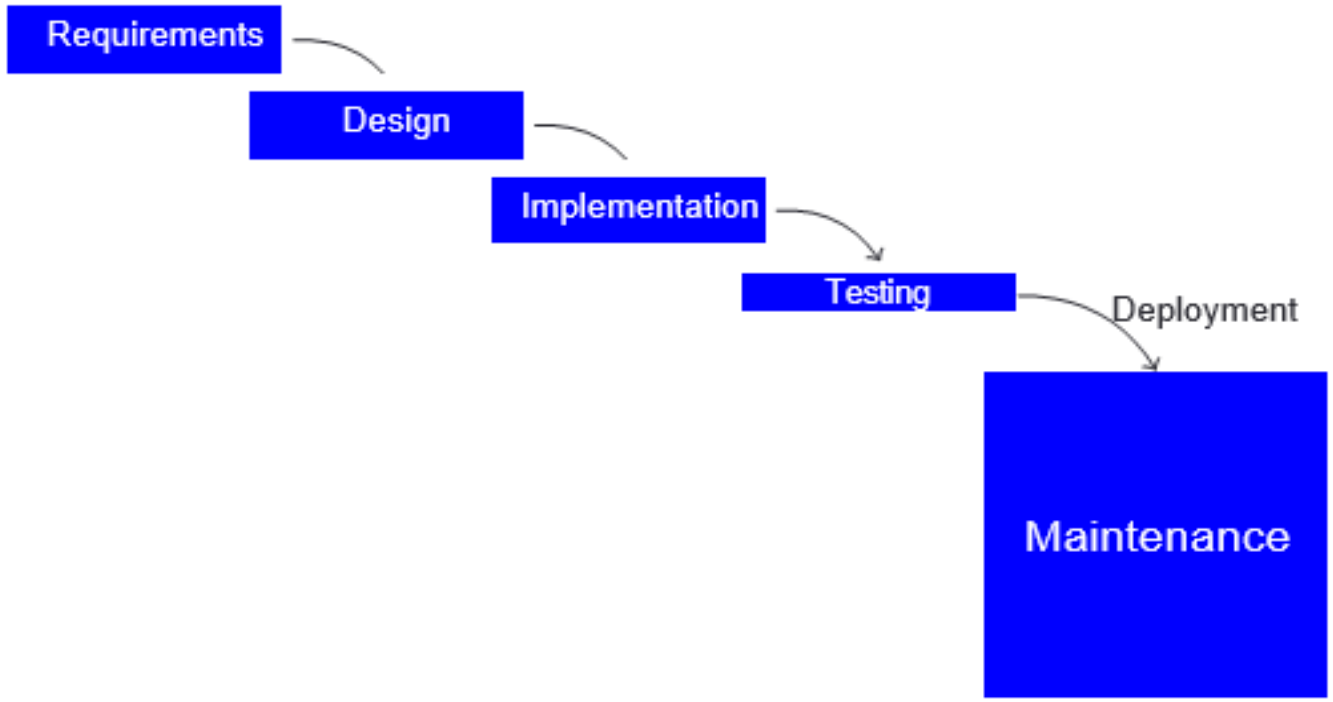
- Carry Out the Plan
  - Does the solution conform to the plan? Is source code traceable to the design model?
  - Is each component part of the solution provably correct? Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Software Engineering for Us

- Examine the Result
  - Is it possible to test each component part of the solution? Has a reasonable testing strategy been implemented?
  - Does the solution produce results that conform to the data, functions, and features that are required? Has the software been validated against all stakeholder requirements?

Software Engineering for Us

- Maintenance accounts for about 70–90% of the total lifecycle budget of a software project [Pigoski, 1996; Seacord et al., 2003]

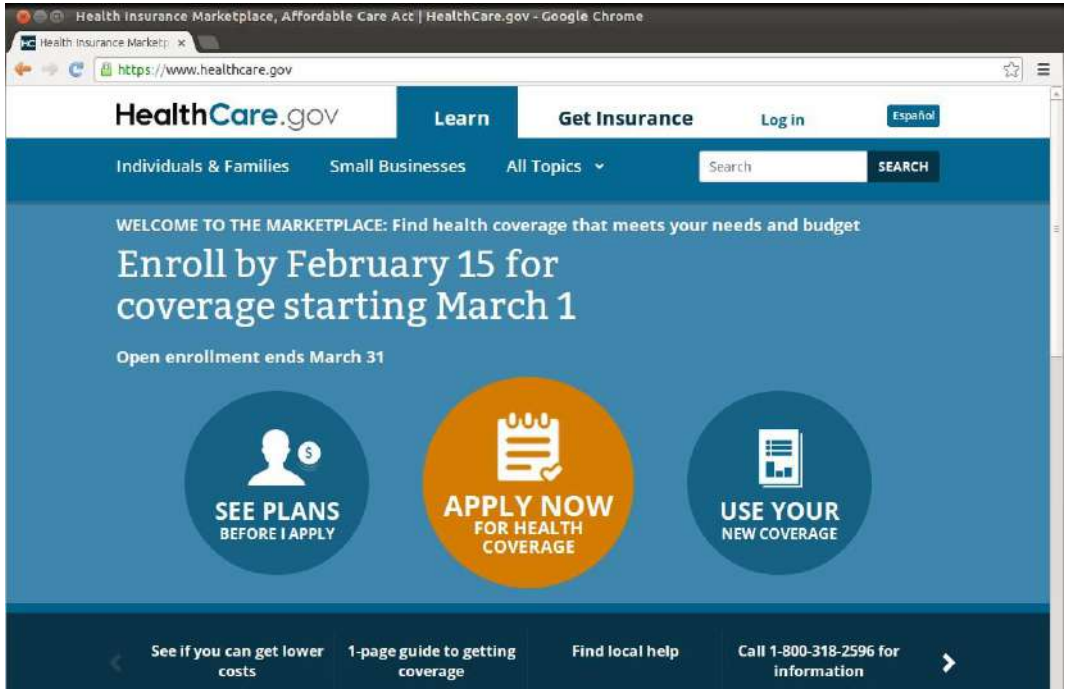


Myths of Software Engineering

- Software myths
  - affect managers, customers (and other non-technical stakeholders) and practitioners and
  - are believable because they often have elements of truth, but
  - invariably lead to bad decisions
  - therefore insist on reality as you navigate your way through software engineering

Myths of Software Engineering

- Management Myths
  - If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).
  - If I decide to outsource the software project to a third party, I can just relax and let that firm build it.



Myths of Software Engineering

- Customer Myths
  - A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.
  - Software requirements continually change, but change can be easily accommodated because software is flexible.

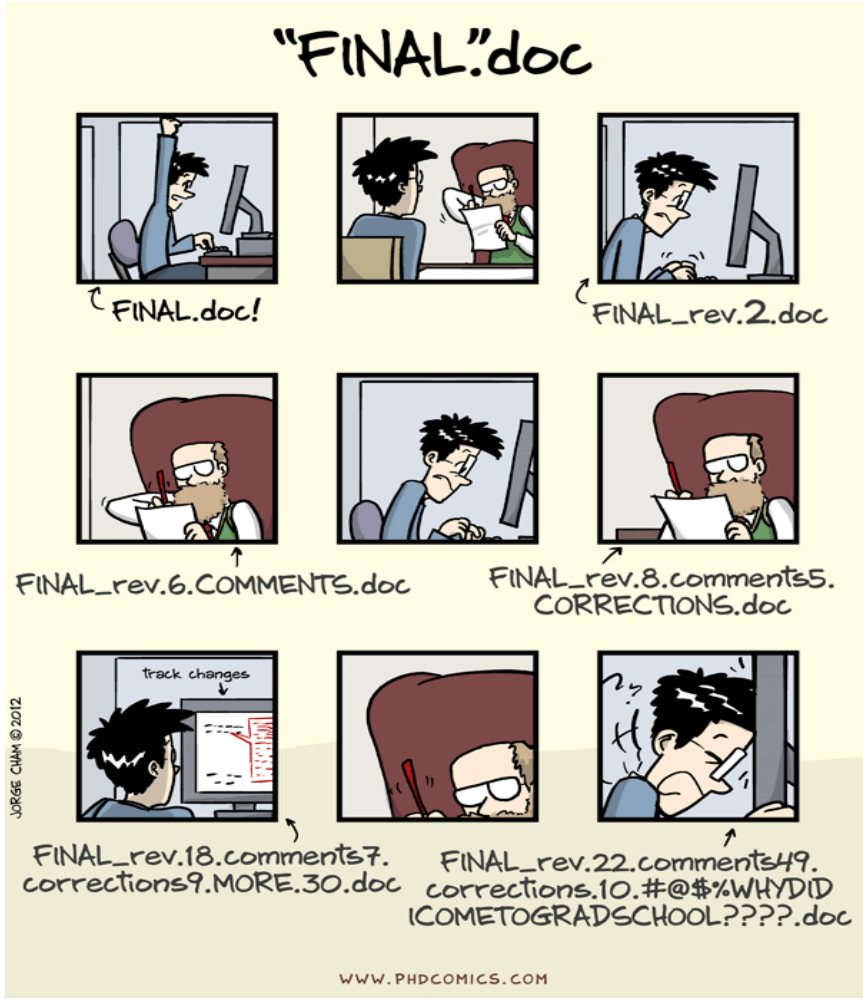
Myths of Software Engineering

- Practitioner's Myths
  - Once we write the program and get it to work, our job is done.
  - Until I get the program "running" I have no way of assessing its quality.
  - The only deliverable work product for a successful project is the working program.
  - Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

Image source: https://commons.wikimedia.org/wiki/File:Städtische_Bücherei_Radstadt_-_book_tower_detail.jpg

Software Engineering: Version Control

- The "First Law"
  - No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle. (Bersoff et al., 1980; emphasis added)
- Why Version Control?



"Piled Higher and Deeper" by Jorge Cham www.phdcomics.com

Software Engineering: Version Control

- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process
- A version control system implements or is directly integrated with four major capabilities:
  - a project database (repository) that stores all relevant configuration objects;
  - a version management capability that stores all versions of a configuration object (or enables any version to be constructed using differences from past versions);
  - a make facility that enables the software engineer to collect all relevant configuration objects and construct a specific version of the software; and
  - an issues tracking (also called bug tracking) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.

Software Engineering

- Software Configuration Management
  - A repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner
  - The repository performs or precipitates the following functions:
    - Data integrity
    - Information sharing
    - Tool integration
    - Data integration
    - Methodology enforcement
    - Document standardization

Software Engineering: Repository Features

- Versioning: Saves all of these versions to enable effective management of product releases and to permit developers to go back to previous versions.
- Change Management: The repository manages a wide variety of relationships among the data elements stored in it.
- Requirements tracing: Provides the ability to track all the design and construction components and deliverables that result from a specific requirement specification
- Configuration Management: Keeps track of a series of configurations representing specific project milestones or production releases. Version management provides the needed versions, and link management keeps track of interdependencies.
- Audit Trails: Establishes additional information about when, why, and by whom changes are made

605.601 Foundations of Software Engineering

Textbook

- Software Engineering – Ian Sommerville – 10th Edition

Suggested Readings

- Beginning Software Engineering – Rod Stephens
- Software Engineering – A Practitioner's Approach – Roger Pressman and Bruce Maxim
- Software Engineering with UML – Bhuvan Unhelkar
- The Mythical Man Month – Fredrick Brooks, Jr.