Introduction to Software Engineering

(CS350)

Lecture 01

Jongmoon Baik

Class Information

- Instructor: Jongmoon Baik
  - Office: ICT Building B/D #502
  - Phone: 042-350-3556/010-4618-5904
  - Email: jbaik@kaist.ac.kr
  - Office Hour: **MON & WED: 10:30AM-12:00PM** (or By Appointment)
- Class Info.
  - Class Hours: **MON & WED 09:00AM – 10:15AM**
  - Class Room: **ICT Building (N1), Lecture Room 112**
  - We’ll start on time with any questions and end on time

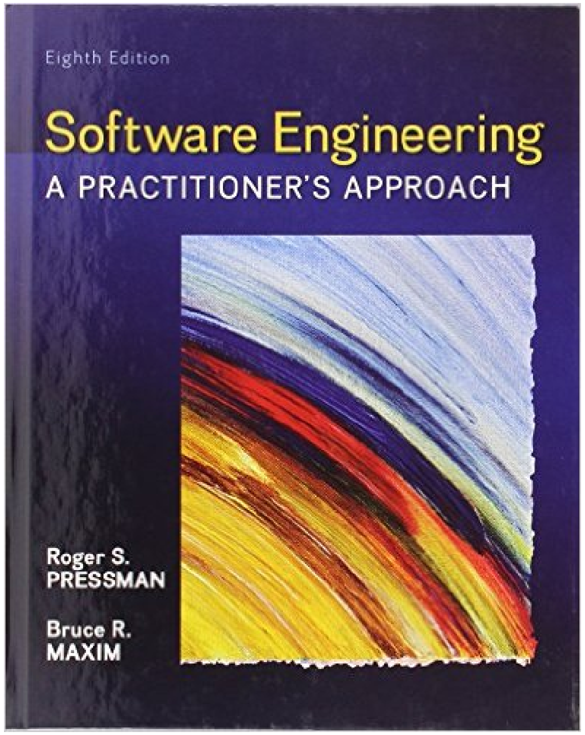
- Teaching Assistant: Jong-In Jang
  - Email: forestar0719@kaist.ac.kr
  - Office: N1, Rm. 525 Office Hours: TBA
  - Tel: 010-350-7756/010-3736-5844

Admin Notes

- Class Website:
  - http://spiral.kaist.ac.kr/wp/2016springcs350/
  - Announcements: You must check periodically
  - All assignments, lecture notes and supplemental materials are available on Class Schedule
- E-Mail
  - Be careful as it does not show other recipients
  - Send e-mails with subject line starting with “[CS350]”

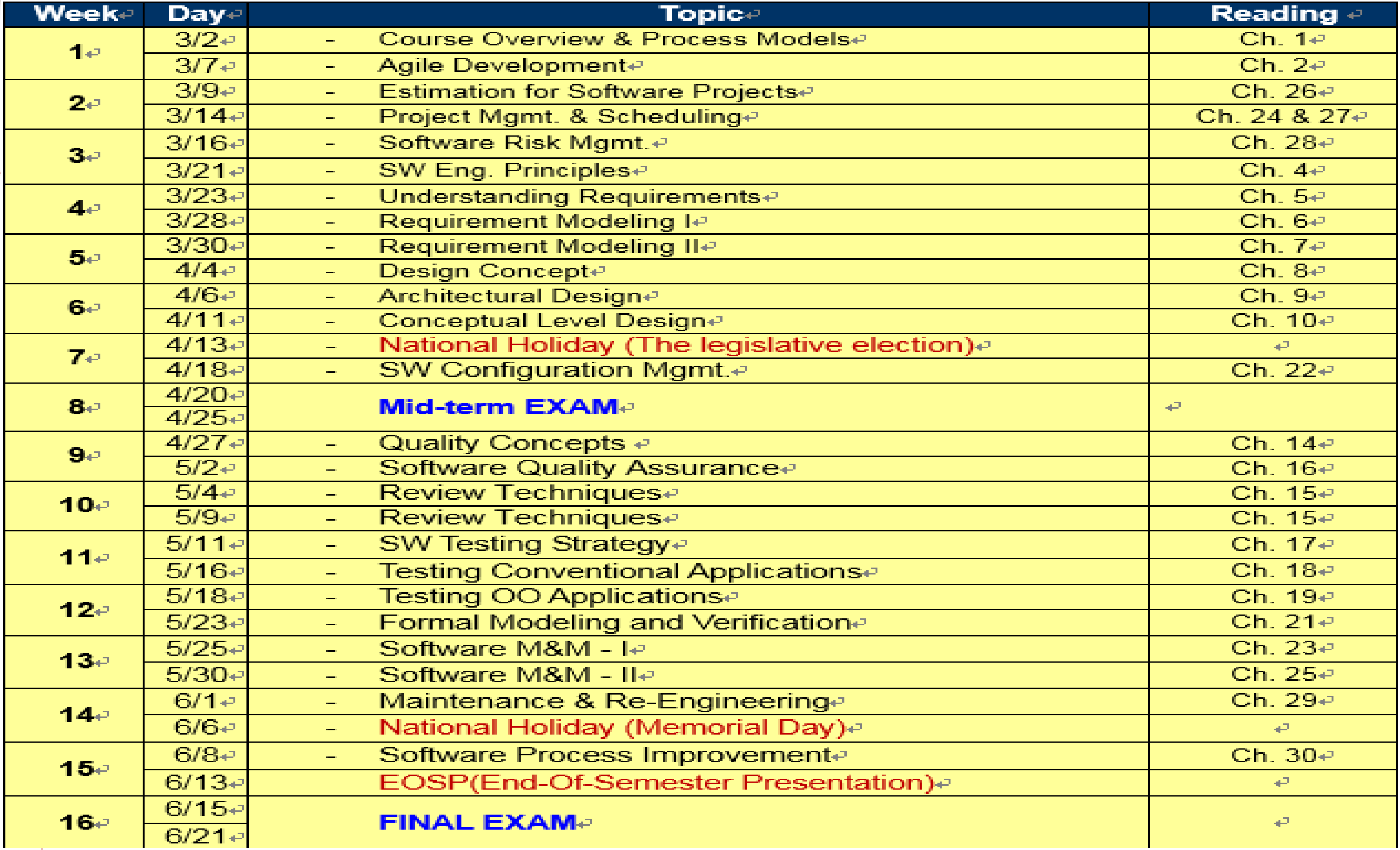
Text Book & References

- Text Book
  - Roger S. Pressman & Bruce Maxim, Software Engineering: A Practitioner's Approach, McGraw-Hill, 8th Edition, ISBN-13: 978-0078022128/_ISBN-10: 0078022126



- References
  - Will be provided on class web site

Class Schedule



Week	Day	Topic	Reading
1	3/2	- Course Overview & Process Models	Ch. 1
	3/7	- Agile Development	Ch. 2
2	3/9	- Estimation for Software Projects	Ch. 26
	3/14	- Project Mgmt. & Scheduling	Ch. 24 & 27
3	3/16	- Software Risk Mgmt.	Ch. 28
	3/21	- SW Eng. Principles	Ch. 4
4	3/23	- Understanding Requirements	Ch. 5
	3/28	- Requirement Modeling I	Ch. 6
5	3/30	- Requirement Modeling II	Ch. 7
	4/4	- Design Concept	Ch. 8
6	4/6	- Architectural Design	Ch. 9
	4/11	- Conceptual Level Design	Ch. 10
7	4/13	- National Holiday (The legislative election)	
	4/18	- SW Configuration Mgmt.	Ch. 22
8	4/20 4/25	**Mid-term EXAM**	
9	4/27	- Quality Concepts	Ch. 14
	5/2	- Software Quality Assurance	Ch. 16
10	5/4	- Review Techniques	Ch. 15
	5/9	- Review Techniques	Ch. 15
11	5/11	- SW Testing Strategy	Ch. 17
	5/16	- Testing Conventional Applications	Ch. 18
12	5/18	- Testing OO Applications	Ch. 19
	5/23	- Formal Modeling and Verification	Ch. 21
13	5/25	- Software M&M - I	Ch. 23
	5/30	- Software M&M - II	Ch. 25
14	6/1	- Maintenance & Re-Engineering	Ch. 29
	6/6	- National Holiday (Memorial Day)	
15	6/8	- Software Process Improvement	Ch. 30
	6/13	EOSP(End-Of-Semester Presentation)	
16	6/15 6/21	**FINAL EXAM**	

Above schedule is subject to change

Grading Policy

- **Exams (40%) – Individual work**
  - Midterm (20%)
  - Final (20%)

- **Term Project (50%) – Group work**
  - Assignment reports (20%)
  - Final project report and presentation (30%)

- **Participation, Attendance & Instructor Judgment (10%)**
  - ✓ *Our perception! Not yours…. Ask if you don't know*
  - ✓ I will call on people randomly at first, not so later
  - ✓ Be proactive, but don't just "run the mouth"
  - ✓ 15% absent – Fail

Assignments

- Each Assignment: Posted on the Web
- Due: At the beginning of the class on the due date
- Submission
  - Hard copy to T.A at the class
  - Email soft copy to T.A. CC to the Instructor
- Late Penalty
  - One day (30%), Two days (50%)
  - Two days after due date: No Acceptance

General Writing Notes

- Must be readable
- PLEASE, 12 pitch font minimum
- Simple font
- 1.5 spacing is nice BUT not mandatory
- Use indentation, bold, etc. as needed
- Spelling and grammar count! (English)
- Must make sense to the reader

“If you don’t like reading it, we won’t either”

Citation of Your Source

- Typically one warning, with a reduced grade
- Then 0’s, or fail in class
- If using material verbatim
  - – Put in quotations with “according to”etc.
  - – I don’t need full source cite
    - According to Jongmoon, “………..”
    - or at end of sentence, paragraph, “whitten et. al., pgs. 47-51”
- If in doubt, ask
  - – Paraphrasing, still state source, but quotations may not be needed

Plagiarism !!!

“The Problem”

- From www.academicintegrity.org, in U.S.
  - 70% of students admit to some cheating
  - 25% admit to cheating on major tests
  - 50% on written assignments in past year
  - 40% to plagiarizing from the internet
  - 77% don’t see this as a “serious issue.”
  - 49% admit to unpermitted student collaboration
  - Faculty reluctance to be “bad guys.”
- Cheating, copying other work, plagiarism is on the rise in US universities.
- Many students feel that they need to “cheat” in order to be competitive
- Some students have stated that “cheating” is acceptable in some cultures
- Some have stated that plagiarism is a form of flattery

What is plagiarism?

- According to the Merriam-Webster Online Dictionary, to "plagiarize" means
  - to steal and pass off (the ideas or words of another) as one's own
  - to use (another's production) without crediting the source
  - to commit literary theft
  - to present as new and original an idea or product derived from an existing source.
- An act of fraud (stealing someone else's work)

The Solution

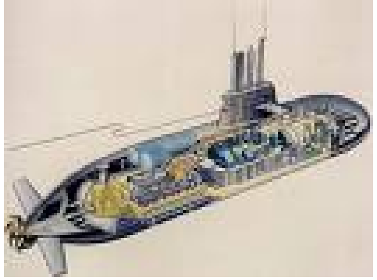
- Professional integrity
- Unlike study, faculty here have no problem dealing with Plagiarism/Cheating
  - No greater offense
  - Allowing yourself to be copied…
- Reputation will follow you

Software & Software Engineering

What is Software?

Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information and (3) documentation that describes the operation and use of the programs.

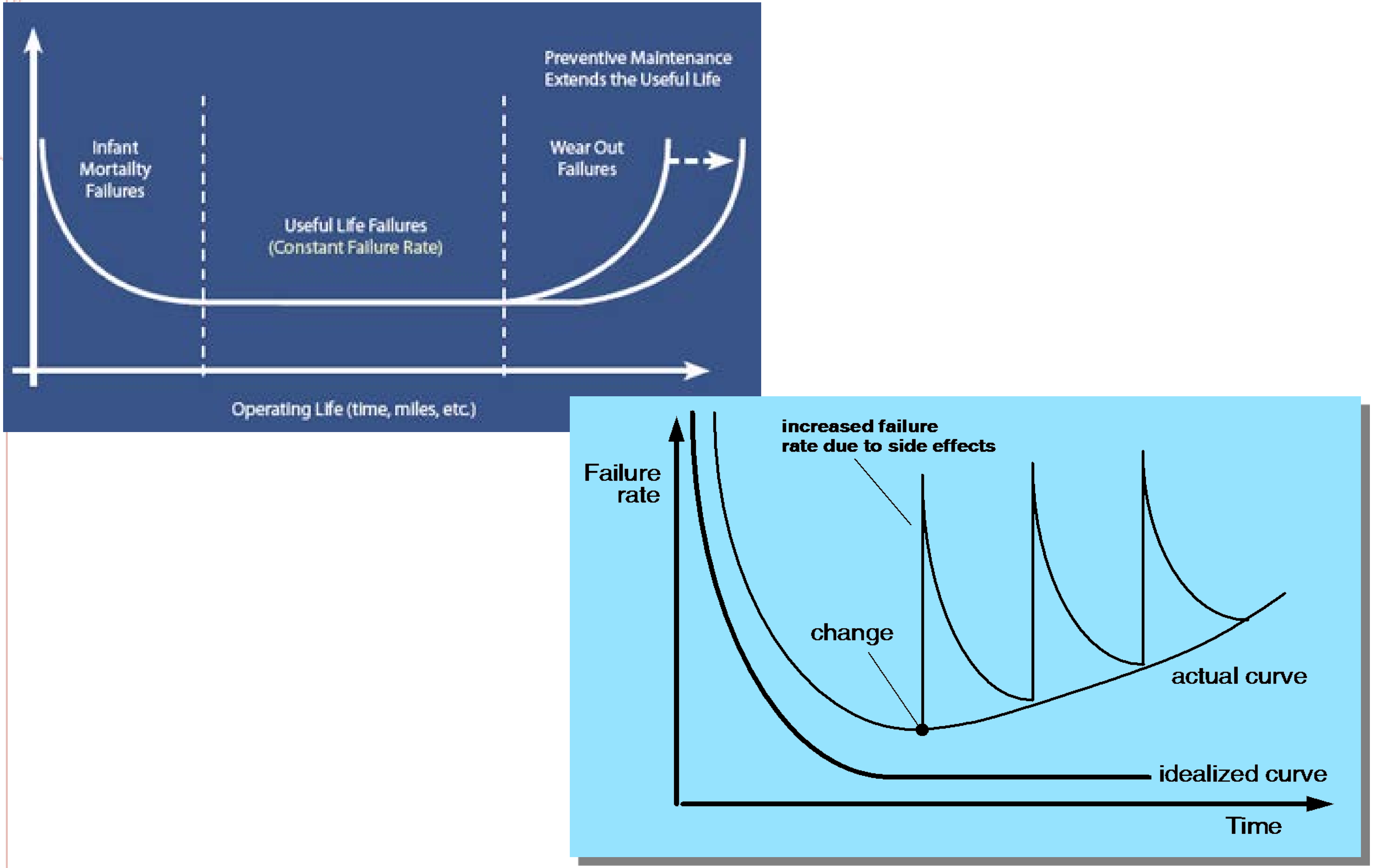
Software is a term primarily used for digitally stored *data* such as computer programs and other kinds of information read and written by computers.



Characteristics of Software

- *Software is a logical rather than a physical system element. (Intangible)*
- *Software is developed or engineered, it is not manufactured in the classical sense.*
- *Software doesn't "wear out."*
  - But, it does deteriorate.
- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

Failure Curves of HW & SW



Software Applications

- System software
- Application software
- Engineering/Scientific software
- Embedded software
- Product-line software
- Web/Mobile applications
- AI software (robotics, neural nets, game playing)

Software—New Categories

- Open world computing—pervasive, distributed computing
- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—”free” source code open to the computing community (a blessing, but also a potential curse!)
- Also … (see Chapter 31)
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

Evolution of Legacy Software

Why must it change?

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

WebApps

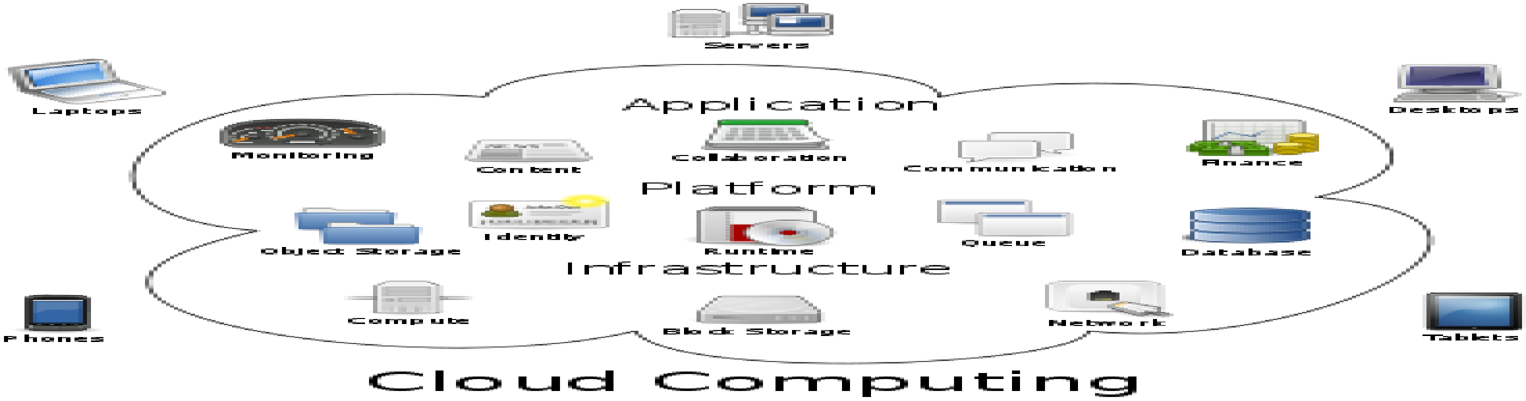
- Modern WebApps are much more than hypertext files with a few pictures
- WebApps are augmented with tools like XML and Java to allow Web engineers including interactive computing capability
- WebApps may standalone capability to end users or may be integrated with corporate databases and business applications
- Semantic web technologies (Web 3.0) have evolved into sophisticated corporate and consumer applications that encompass semantic databases that require web linking, flexible data representation, and application programmer interfaces (API's) for access
- The aesthetic nature of the content remains an important determinant of the quality of a WebApp.

Mobile Apps

- Reside on mobile platforms such as cell phones or tablets
- Contain user interfaces that take both device characteristics and location attributes
- Often provide access to a combination of web-based resources and local device processing and storage capabilities
- Provide persistent storage capabilities within the platform
- A *mobile web application* allows a mobile device to access to web-based content using a browser designed to accommodate the strengths and weaknesses of the mobile platform
- A *mobile app* can gain direct access to the hardware found on the device to provide local processing and storage capabilities
- As time passes these differences will become blurred

Cloud Computing

- *Cloud computing* provides distributed data storage and processing resources to networked computing devices
- Computing resources reside outside the cloud and have access to a variety of resources inside the cloud
- Cloud computing requires developing an architecture containing both frontend and backend services
- Frontend services include the client devices and application software to allow access
- Backend services include servers, data storage, and server-resident applications
- Cloud architectures can be segmented to restrict access to private data



Product Line Software

- *Product line software* is a set of software-intensive systems that share a common set of features and satisfy the needs of a particular market
- These software products are developed using the same application and data architectures using a common core of reusable software components
- A software product line shares a set of assets that include *requirements, architecture, design patterns, reusable components, test cases,* and other work products
- A software product line allow in the development of many products that are engineered by capitalizing on the commonality among all products with in the product line

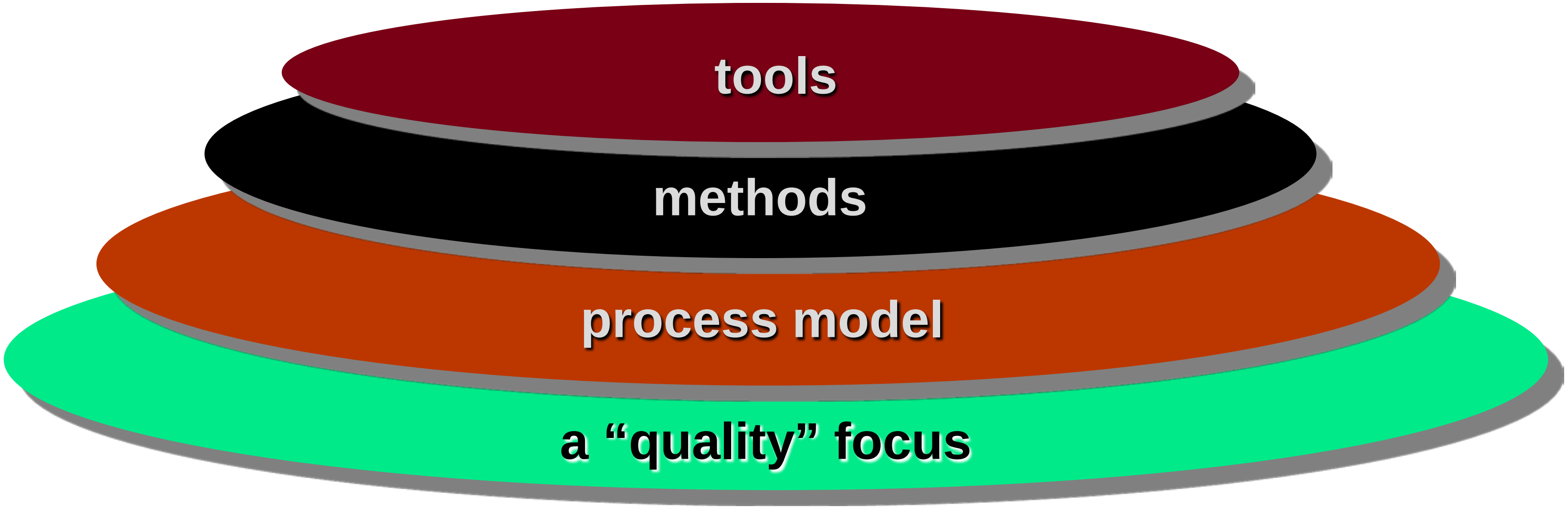
Software Engineering

- Some realities:
  - *a concerted effort should be made to understand the problem before a software solution is developed*
  - *design becomes a pivotal activity*
  - *software should exhibit high quality*
  - *software should be maintainable*

- *[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. – Fritz Bauer*

- The IEEE definition:
  - *Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).*

A Layered Technology



Software Engineering

A Process Framework

Process framework

- Framework activities
  - work tasks
  - work products
  - milestones & deliverables
  - QA checkpoints
- Umbrella Activities

Activities for A Generic Process Framework

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities

- Software project tracking and control
- Risk management
- Software quality assurance
- Technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

Adapting a Process Model

- the overall flow of activities, actions, and tasks and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail and rigor with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

The Essence of Practice

- George Polya 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).

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?

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 sub-problems be defined?*
  - If so, are solutions readily apparent for the sub-problems?
- *Can you represent a solution in a manner that leads to effective implementation?*
  - Can a design model be created?

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?

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?

Hooker's General Principles

- 1: *The Reason It All Exists*
- 2: *KISS (Keep It Simple, Stupid!)*
- 3: *Maintain the Vision*
- 4: *What You Produce, Others Will Consume*
- 5: *Be Open to the Future*
- 6: *Plan Ahead for Reuse*
- *7: Think!*

Software Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- 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

Management Myths

- *We already have a book that’s full of standards and procedures for building software. Won’t that provide my people with everything they need to know?*
- *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.,*

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.*

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 creative voluminous and unnecessary documentation and will iteratively slow us down.*

1st Assignment

- Fill out the questionnaire and submit it to T.A. before the next class.

- Organize a team (3-4 people) for your team project by MAR. 11 (FRI)
  - Send team members’ information with your team name to T.A. (CC to the instructor)

Q & A