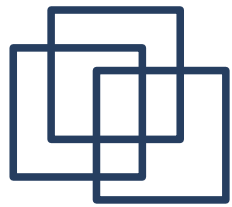


CMSC 128

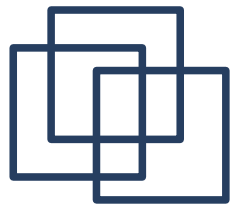
Introduction to Software Engineering Second Semester AY 2009-2010

jachermocilla@uplb.edu.ph



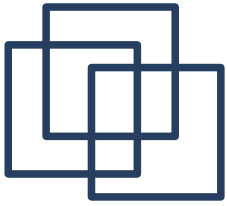
Product and Process

- What is computer software?
- Why do we struggle to build high quality computer-based systems?
- How can we categorize application domains for computer software?
- What myths about software still exists?
- What is a 'software process'?
- Is there a generic way to assess the quality of a process?



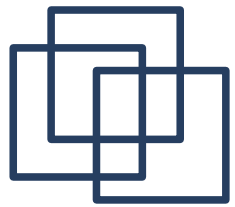
Product and Process(2)

- What process models can be applied to software development?
- How do linear and iterative process models differ?
- What are their strengths and weaknesses?
- What advanced process models have been proposed for software engineering work?



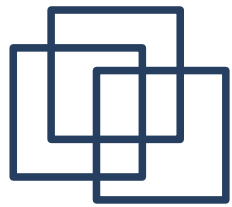
The Product

- Computer software has become a driving force
 - Business decision-making
 - Scientific investigation
 - Engineering problem solving
 - Embedded: transportation, medical, telecommunications, military, industrial processes, entertainment, office products
 - ..almost endless



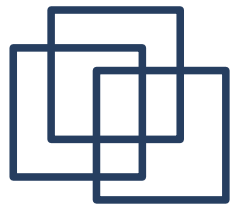
Evolving Role of Software

- Software: *Product* and *Vehicle* for *delivering a product*
- Product
 - information transformer
- Vehicle for delivering a product
 - control of computer (Operating Systems)
 - communication of information (Networks)
 - creation and control of other programs (Software Tools and Environments)



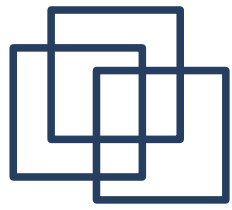
Evolving Role of Software(2)

- Role of software has undergone significant change
 - improvements in hardware
 - computing architectures
 - increase in memory and storage capacity
 - variety of exotic input and output options
- Resulted to sophisticated and complex computer-based systems
- Poses huge problems for builders



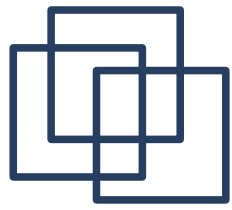
Evolving Role of Software(3)

- Early Years (1950's)
 - Batch orientation
 - Limited Distribution
 - Custom Software
- Second Era (Mid 1960's-Late 1970's)
 - Multiuser
 - Real-time
 - Database
 - Product software



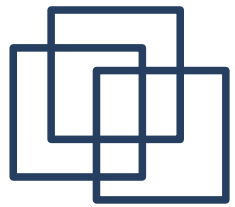
Evolving Role of Software(4)

- Third Era (Mid 1970's)
 - Distributed Systems
 - Embedded “intelligence”
 - microprocessor
 - Low cost hardware
 - personal computer
 - Consumer impact



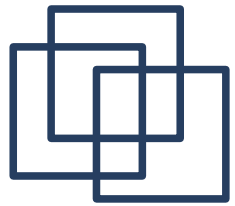
Evolving Role of Software(5)

- Fourth Era (Late 1980's)
 - Powerful desktop systems
 - Object-oriented technologies
 - Expert Systems
 - Artificial neural networks
 - Parallel Computing
 - Networks
 - The Internet



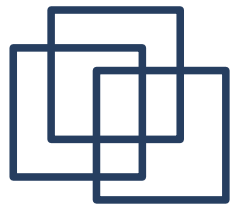
Evolving Role of Software(6)

- Software related problems persisted
 1. Hardware advances outpace ability to build software to tap hardware power
 2. Ability to build programs cannot keep pace with demand for new business and market needs
 3. Continuing struggle to increase reliability and quality of software
 4. Difficulty in supporting and enhancing existing programs



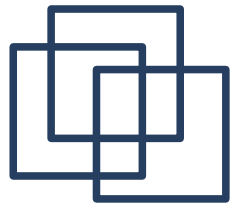
Industry Perspective

- Early years
 - Hardware oriented
 - hardware engineering
 - Programming is an “art form”
 - few formal methods exist, few use formal methods
 - trial and error
 - undisciplined
- Today
 - Software rules!



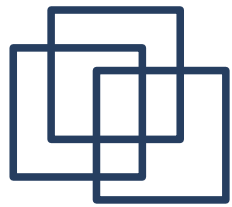
Industry Perspective (2)

- Questions asked by managers and technical practitioners
 - Why does it take so long to get programs finished?
 - Why are costs so high?
 - Why can't we find all errors before we give the software to our customers?
 - Why do we have difficulty in measuring progress as software is being developed?



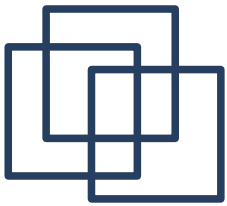
Aging Software Plant

- Information system applications written twenty years ago are virtually unmaintainable
- Internal structure of engineering applications used to produce critical design data are hard to understand
- Embedded systems that exhibit strange and sometimes unexplained behavior cannot be taken out of service
 - no available replacement is available



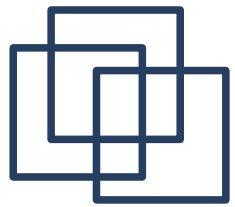
Software Competitiveness

- Software developers in companies were once the only show in town
 - programs are custom-built
 - they dictated costs, schedule, and quality
- Today, software is an extremely competitive business
 - off-the-shelf
 - outsourcing
 - cost, timeliness, and quality are primary drivers



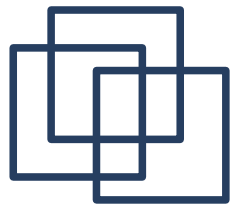
Software

- Textbook descriptions
 - instructions (computer programs) that when executed provide desired function and performance
 - data structures that enable the programs to adequately manipulate information
 - documents that describe the operation and use of the programs
- Descriptions above are not enough to really understand what it means



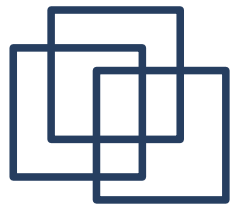
Software Characteristics

- Software is a logical rather than a physical system element
- Software is developed or engineered, not manufactured in the classical sense
 - high quality is dependent on good design
- Software doesn't “wear out”
 - software deteriorates (due to changes)
 - no software spare parts



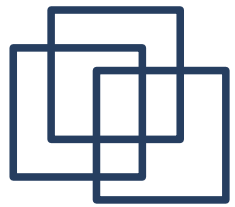
Software Characteristics(2)

- Most software is custom-built, rather than being assembled from existing components
 - slowly changing because of object-oriented technologies
 - Libraries (API)
 - Plugin Architecture



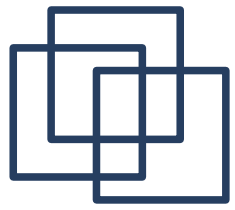
Software Components

- Collection of standard components is typical of an engineering discipline
 - realized in hardware
 - still to be achieved in software
- Reusability is an essential characteristic of a high quality software component
 - algorithm reuse (functions)
 - algorithm+data structure (objects)
 - GUI



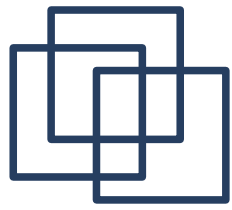
Software Applications

- Software may be applied in any situation for which a set of procedural steps (i.e., an algorithm) has been defined
 - Nature of software can be determined by information *content* and *determinacy*
- Content
 - meaning and form of incoming and outgoing information
- Determinacy
 - predictability of order/timing of information



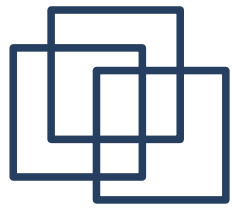
Software Applications (2)

- System Software
 - written to service other programs
 - ex. compilers, operating systems, drivers
 - characterized by heavy interactions with hardware; heavy usage by multiple users; concurrent operations that require scheduling; resource sharing; process management; complex data structures; and multiple external interfaces



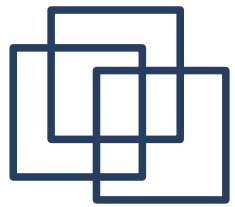
Software Applications (3)

- Real-Time Software
 - monitor/analyze/control real world events as they occur
 - ex. missile detection system, submarine navigation system
 - includes data gathering, analysis, control/output, and monitoring components
 - must respond under strict time constraints



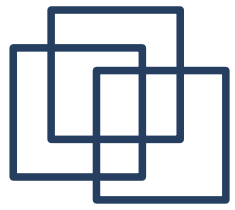
Software Applications (4)

- Business Software
 - largest single software application area
 - ex. payroll, inventory, POS, MIS
 - restructure existing data to support business operations or management decision making
 - probably what you will be working on in the industry



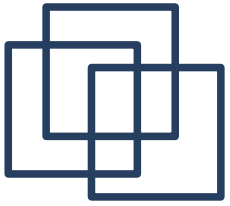
Software Applications (5)

- Engineering and Scientific Software
 - number-crunching
 - ex. stress analysis systems for automotives, space shuttle orbital dynamics, CAD software, YACAS?
- Embedded Software
 - used in intelligent products
 - ex. brake systems in car, cellphones
 - normally resides in ROM



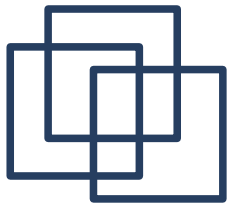
Software Applications (6)

- Personal Computer Software
 - desktop applications
 - ex. word processors, spreadsheets, DBMS
 - Commercial-Off-The-Shelf (COTS)
 - several competing products
- Artificial Intelligence Software
 - nonnumerical
 - ex. pattern recognition, game playing
 - expert systems, neural networks, MAS



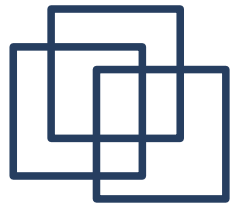
Software Crisis

- Problems will be encountered in the development of computer software
- The problems encompasses not just non-functioning software but also how we develop software, how we maintain existing software, and how we expect to keep pace with growing demand for software



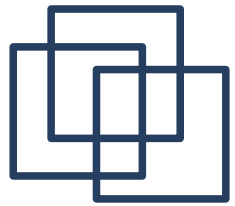
Software Myths

- Misleading attitudes that have caused serious problems for managers and technical people
- Management Myths
 1. Everything my people need to know are in our book of standards and procedures for building software
 2. My people have the state-of-the-art software development tools
 3. If behind schedule, we can add more people



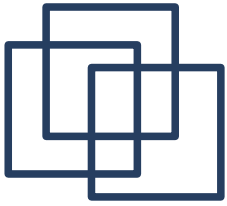
Software Myths (2)

- Customer Myths
 1. A general statement of objectives is enough to begin writing programs-details will come later
 2. Project requirements continually change, but change can be easily accommodated because software is flexible



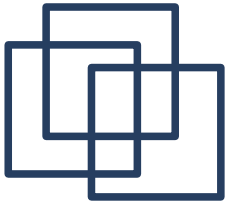
Software Myths (3)

- Practitioner's Myth's
 - Once we write the program and get it to work, our job is done
 - Until I get the program “running“, I really have no way of assessing its quality
 - The only deliverable for a successful project is the working program



Summary

- Software is a key element in the evolution of computer-based systems
- Software has evolved from being a tool to an industry itself
- Software has become a limiting factor in the evolution of computer-based systems
- Software is composed of programs, data, and documentation
- SE aim for creating high quality software



Reference

- Roger S. Pressman. Software Engineering: A Practitioner's Approach, 4th Ed. McGraw-Hill, 1997. Chapter 1.