

# Advanced Software Engineering

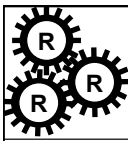
**Dr. Cheng**

## Overview of Software Engineering and Development Processes

CSE870  
(Spring 2012)

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering

1

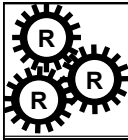


## Instructor Background

- Professor in CSE
- Areas of interest: SE, formal/informal approaches to SE, model-driven engineering, dynamically adaptive systems, evolutionary computation -- all applied to high assurance systems
- Joined MSU in 1990
- Bachelors degree from Northwestern University
- MS and PhD in CS from University of Illinois (UIUC)
- Co-founder of Software Engineering and Network Systems (SENS) Lab
  - 5 faculty members
  - 25-30 RAs, postdocs
- Also a member of DevoLab; KT Manager BEACON
- Research currently funded by NSF, ARO, and industry.
  - Recent projects with ONR, AFRL, NASA

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering

2



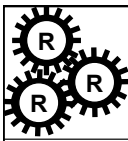
## What is Software Engineering?

- Systematic approach for developing software
- Methods and techniques to develop and maintain quality software to solve problems.

(Software Engineering: Methods and Management, Pfleeger, 1990)

- Study of the *principles* and *methodologies* for developing and maintaining software systems.

("Perspectives on Software Engineering", Zelkowitz, 1978)



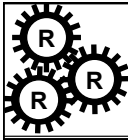
## What is Software Engineering?

- *Practical* application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them.

("Software Engineering", Boehm, 1976)

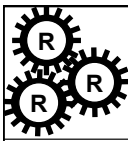
- Deals with establishment of sound engineering principles and methods in order to *economically* obtain software that is reliable and works on real machines.

("Software Engineering", Bauer, 1972)



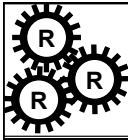
## Questions addressed by Software Engineering

- How do we ensure the quality of the software that we produce?
- How do we meet growing demand and still maintain budget control?
- How do we avoid disastrous time delays?



## Why apply Software Engineering to Systems?

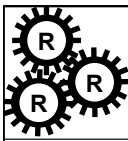
- Provide an understandable process for system development.
- Develop systems and software that are maintainable and easily changed.
- Develop robust software and system.
- Allow the process of creating computing-based systems to be repeatable and manageable.



## Objectives of Course

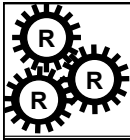
- Provide exposure to leading-edge topics
  - Emphasize object-orientation and modeling
  - Emphasize requirements and design
  - Emphasize assurance of computing-based systems
- Provide hands-on experience to reinforce concepts
  - Homework assignments
  - Modeling and specification assignments
- Synthesize several topics into mini-projects
  - Programming/design project with written component
  - Prepare presentation materials for lay audience.
- **Overarching application theme: onboard electronics in automotive systems**

7



## Tentative Topics

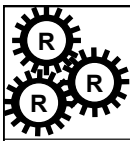
- Requirements Engineering
- Unified Modeling Language (UML)
- Architectural Styles
- Design Patterns
- Security
- Aspect-Oriented Programming
- Embedded Systems



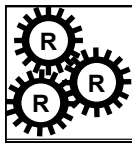
## Administrative Work

- Background Survey
- Initial Assessment
- Tentative Evaluation Mechanisms:

Exams (2)	40 %
Homework/Design Exercises	25 %
Mini-Project(s)	35 %

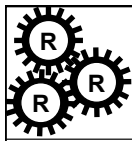


STOP



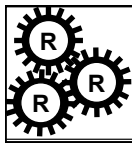
## Software Engineering Phases

- Definition: What?
- Development: How?
- Maintenance: Managing change
- Umbrella Activities: Throughout lifecycle



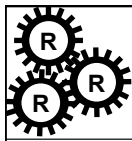
## Definition

- Requirements definition and analysis
  - Developer must understand
    - Application domain
    - Required functionality
    - Required performance
    - User interface



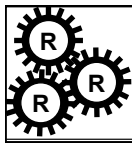
## Definition (cont.)

- Project planning
  - Allocate resources
  - Estimate costs
  - Define work tasks
  - Define schedule
- System analysis
  - Allocate system resources to
    - Hardware
    - Software
    - Users



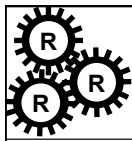
## Development

- Software design
  - User interface design
  - High-level design
    - Define modular components
    - Define major data structures
  - Detailed design
    - Define algorithms and procedural detail



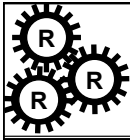
## Development (cont.)

- Coding
  - Develop code for each module
  - Unit testing
- Integration
  - Combine modules
  - System testing



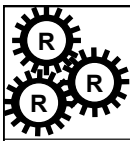
## Maintenance

- Correction - Fix software defects
- Adaptation - Accommodate changes
  - New hardware
  - New company policies
- Enhancement - Add functionality
- Prevention - make more maintainable



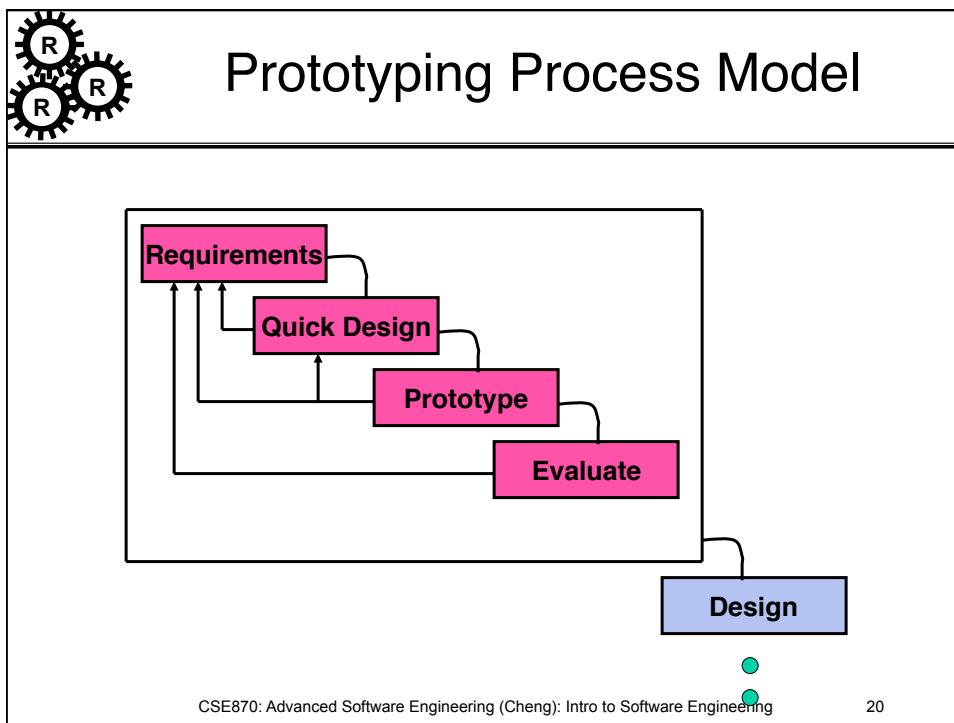
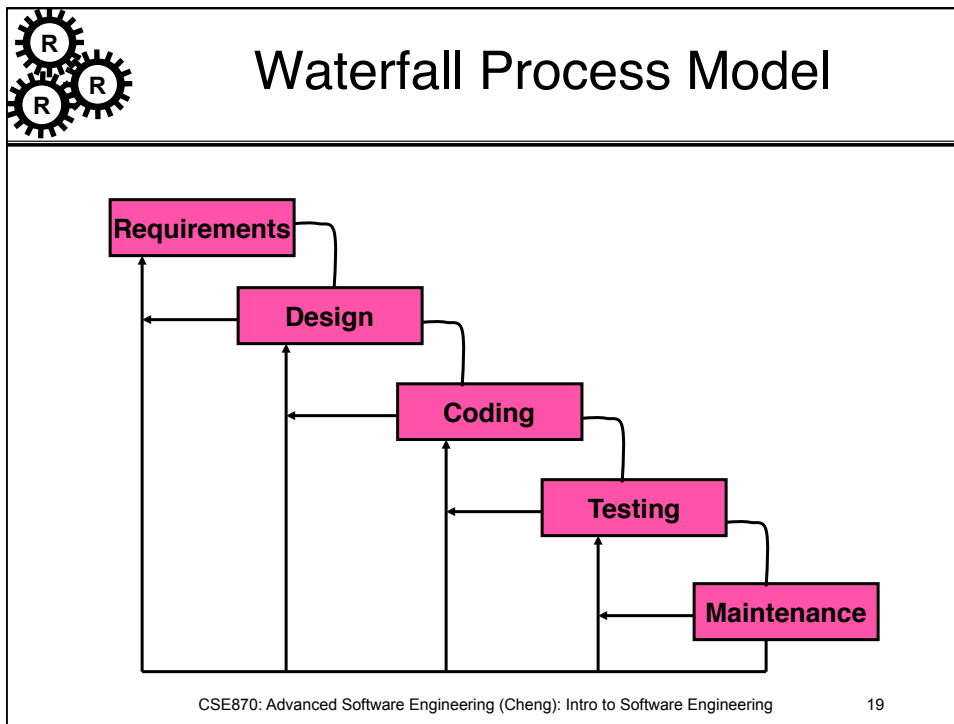
## Umbrella Activities

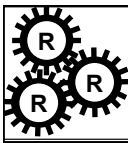
- Reviews - assure quality
- Documentation - improve maintainability
- Version control - track changes
- Configuration management - integrity of collection of components



## Development Process

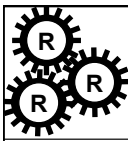
- Step-by-step procedure to develop software
- Typically involves the major phases:
  - analysis
  - design
  - coding
  - testing



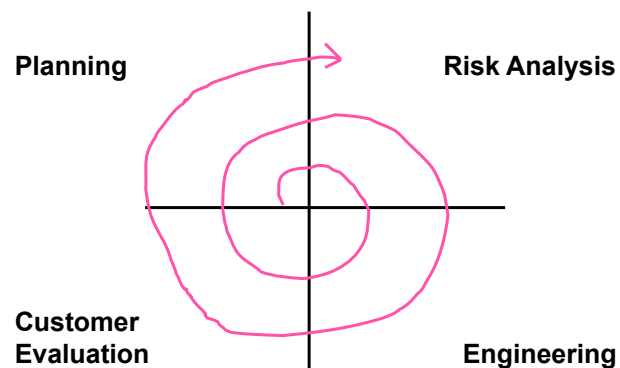


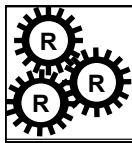
## When to use prototyping?

- Help the customer pin down the requirements
  - Concrete model to “test out”
  - Often done via the user interface
- Explore alternative solutions to a troublesome component
  - e.g., determine if an approach gives acceptable performance
- Improve morale
  - Partially running system provides visibility into a project



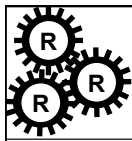
## Spiral Process Model





## Process Models


- Idealized views of the process
- Different models are often used for different subprocesses
  - may use spiral model for overall development
    - prototyping for a particularly complex component
    - waterfall model for other components



## Capability Maturity Model

- Level 1: Initial
  - ad hoc
  - success depends on people
- Level 2: Repeatable
  - track cost, schedule, functionality
- Level 3: Defined
  - use standardized processes
- Level 4: Managed
  - collect detailed metrics
- Level 5: Optimizing
  - continuous process improvement
  - “built-in” process improvement


Software Engineering Institute:  
<http://www.sei.cmu.edu/cmm/>



## Why is software development so difficult?

- **Communication**
  - Between customer and developer
    - Poor problem definition is largest cause of failed software projects
  - Within development team
    - More people = more communication
    - New programmers need training
- **Project characteristics**
  - Novelty
  - Changing requirements
    - 5 x cost during development
    - up to 100 x cost during maintenance
  - Hardware/software configuration
  - Security requirements
  - Real time requirements
  - Reliability requirements

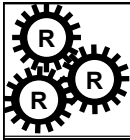
CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering 25



## Why is software development difficult? (cont.)

- **Personnel characteristics**
  - Ability
  - Prior experience
  - Communication skills
  - Team cooperation
  - Training
- **Facilities and resources**
  - Identification
  - Acquisition
- **Management issues**
  - Realistic goals
  - Cost estimation
  - Scheduling
  - Resource allocation
  - Quality assurance
  - Version control
  - Contracts

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering 26

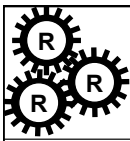


## Summary

- Software lifecycle consists of
  - Definition (what)
  - Development (how)
  - Maintenance (change)
- Different process models concentrate on different aspects
  - Waterfall model: maintainability
  - Prototype model: clarifying requirements
  - Spiral model: identifying risk
- Maintenance costs much more than development

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering

27



## Bottom Line

- U.S. software is a major part of our societal infrastructure
  - Costs upwards of \$200 billion/year
- Need to
  - Improve software quality
  - Reduce software costs/risks

CSE870: Advanced Software Engineering (Cheng): Intro to Software Engineering

28