What is Software Engineering
Where Does the Software Engineer Fit In?

• **Computer science**: focusing on computer hardware, compilers, operating systems, and programming languages

• **Software engineering**: a discipline that uses computer and software technologies as problem-solving tools
What is Software Engineering
Where Does the SW Engineer Fit in? (continued)

• Relationship between computer science and software engineering
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
    0 Hardware
    0 Software
    0 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
    - Change in requirements
    - Change in resources

- **Prevention** - make more maintainable

- **Enhancement** - Add functionality
Software Engineering Costs
This is why software process pays off.
open_time = (day == 'Sunday' || 'Saturday') ? 12 : 9;

vs.

if (day == 'Sunday')
    open_time = 12;
    close_time = 6;
else if ( day == 'Saturday')
    open_time = 12;
    close_time = 9;
else
    open_time = 9;
Umbrella Activities

- **Reviews** - assure quality
- **Documentation** - improve maintainability
- **Version control** - track changes
- **Configuration management** - integrity of collection of components
What is Good Software?

- Good software engineering must always include a strategy for producing quality software
- Three ways of considering quality
  - The quality of the product
  - The quality of the process
  - The quality of the product in the context of the business environment
What Is Good Software?
The Quality of the Product

• Users judge external characteristics
  – (e.g., correct functionality, number of failures, type of failures, ease of use)
• Designers and maintainers judge internal characteristics (e.g., types of faults, ease of change)
• Thus different stakeholders may have different criteria
• Need quality models to relate the user’s
1.3 What Is Good Software? The Quality of the Process

- Quality of the development and maintenance process is as important as the product quality

- The development process needs to be modeled

- Modeling will address questions such as
  - Where to find a particular kind of fault
  - How to find faults early
  - How to build in fault tolerance
  - What are alternative activities
1.3 What Is Good Software? The Quality of the Process (continued)

- Models for process improvement
  - SEI’s Capability Maturity Model (CMM)
  - ISO 9000
  - Software Process Improvement and Capability Determination (SPICE)
Review

• What are two complementary strategies for SE-based problem solving?
• What are three ways to consider software quality?
• Who are the three key stakeholders in a software development project?
1.3 What Is Good Software?
The Quality in the Context of the Business Environment

- Business value is as important as technical value
- Business value (in relationship to technical value) must be quantified
- A common approach: return on investment (ROI) – what is given up for other purposes
- ROI is interpreted in different terms: reducing costs, predicting savings, improving productivity, and costs (efforts and resources)
1.9 Information Systems Example

Piccadilly System

• Piccadilly Television: regional British TV franchise

• Advertising scheme has many constraints:
  - alcohol adverts only after 9 pm
  - if actor in show, no same actor in advert within 45 minutes
  - if advert in a class of products, no other advert in same class during same break
  - rates dependent on amount of time bought

• Software to determine, track advertising time
1.9 Information Systems Example
Piccadilly System (continued)

- Piccadilly Television franchise area
1.9 Information Systems Example
Piccadilly System (continued)

• Piccadilly system’s context diagram
1.10 Real Time Example

- Ariane-5 rocket, from the European Space Agency
- June 4, 1996: functioned well for 40 seconds, then veered off course and was destroyed
- Contained four satellites: cost was $500 million
- Reused code from Ariane-4 rocket
1.10 Real Time Example

Ariane-5 Definition of Quality

• From the Lions et al report:
  - “… demonstrated the high quality of the Ariane-5 programme as regards engineering work in general and completeness and traceability of documents.”
  - “… the supplier of the SRI … was only following the specification given to it. … The exception which occurred was not due to random failure but a design error.”
1.11 What this Chapter Means for You

• Given a problem to solve
  – Analyze it
  – Synthesize a solution
• Understand that requirements may change
• Must view quality from several different perspectives
• Use fundamental software engineering concepts (e.g., abstractions and measurements)
2.1 The Meaning of Process

• A **process**: a series of steps involving activities, constraints, and resources that produce an intended output of some kind

• A process involves a set of tools and techniques
2.1 The Meaning of Process

Process Characteristics

• Prescribes all major process activities

• Input:
  - Uses resources (e.g., customer input, specifications),
  - subject to set of constraints (such as schedule, platform reqts)

• Output: Produces intermediate and final products (e.g., models)
2.1 The Meaning of Process

Process Characteristics (continued)

• Structure:
  - May be composed of subprocesses
  - with hierarchy or links

• Properties:
  - Each process activity has entry and exit criteria
  - Activities are organized in sequence, so timing is clear
  - Each process guiding principles, including goals of each activity
  - Constraints may apply to an activity, resource or product
2.1 The Meaning of Process

The Importance of Processes

• Impose consistency and structure on a set of activities

• Guide us to understand, control, examine, and improve the activities

• Enable us to capture our experiences and pass them along
2.2 Software Process Models
Reasons for Modeling a Process

• To form a common understanding
• To find inconsistencies, redundancies, omissions
• To find and evaluate appropriate activities for reaching process goals
• To tailor a general process for a particular situation in which it will be used
2.2 Software Process Models

Software Life Cycle

• When a process involves building a software, the process may be referred to as software life cycle
  - Requirements analysis and definition
  - System (architecture) design
  - Program (detailed/procedural) design
  - Writing programs (coding/implementation)
  - Testing: unit, integration, system
  - System delivery (deployment)
  - Maintenance
2.2 Software Process Models
Software Development Process Models

- Waterfall model
- V model
- Prototyping model
- Operational specification
- Transformational model
- Phased development: increments and iteration
- Spiral model
- Agile methods
2.2 Software Process Models

Waterfall Model

• One of the first process development models proposed
• Works for well understood problems with minimal or no changes in the requirements
• Simple and easy to explain to customers
• It presents
  - a very high-level view of the development process
  - sequence of process activities
• Each major phase is marked by milestones and deliverables (artifacts)
2.2 Software Process Models

Waterfall Model (continued)

- REQUIREMENTS ANALYSIS
- SYSTEM DESIGN
- PROGRAM DESIGN
- CODING
- UNIT & INTEGRATION TESTING
- SYSTEM TESTING
- ACCEPTANCE TESTING
- OPERATION & MAINTENANCE
2.2 Software Process Models

Waterfall Model (continued)

- There is no iteration in waterfall model
- Most software developments apply a great many iterations
Drawbacks of The Waterfall Model

• Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
• Views software development as manufacturing process rather than as creative process
• There are no iterative activities that lead to creating a final product
• Long wait before a final product
2.2 Software Process Models
Waterfall Model with Prototype

• Different types of prototype:
  - **Requirements**: User interface shell to illustrate observable behavior (i.e., used to elicit reqts info)
  - **Design**: A prototype can be a partially developed product (black box design details and/or components)

• Prototyping helps
  - users understand what the system will be like (user interface prototype)
  - developers assess alternative design strategies (design prototype)

• Prototyping is useful for verification and validation
2.2 Software Process Models
Waterfall Model with Prototype (continued)

- Waterfall model with prototyping
2.2 Software Process Models

V Model (continued)
2.2 Software Process Models

V Model

- A variation of the waterfall model
- Uses unit testing to verify procedural design
- Uses integration testing to verify architectural (system) design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted
2.2 Software Process Models

V Model (continued)

- REQUIREMENTS ANALYSIS
- SYSTEM DESIGN
- PROGRAM DESIGN
- CODING
- UNIT & INTEGRATION TESTING
- SYSTEM TESTING
- ACCEPTANCE TESTING
- OPERATION & MAINTENANCE

Arrows indicate the flow of activities and dependencies between the stages.
2.2 Software Process Models

Prototyping Model

- Allows repeated investigation of the requirements or design
- Reduces risk and uncertainty in the development