Chapter 2 Objectives

- What we mean by a “process”
- Software development products, processes, and resources
- Several models of the software development process
- Tools and techniques for process modeling
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)
• 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)
2.2 Software Process Models
Waterfall Model (continued)

- There is no iteration in waterfall model
- Most software developments apply a great many iterations
Sidebar 2.1 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 is 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)

- REQUIREMENTS ANALYSIS → SYSTEM DESIGN
  - Validate requirements
  - Verify design

- SYSTEM DESIGN
  - OPERATION & MAINTENANCE
  - ACCEPTANCE TESTING
  - SYSTEM TESTING
  - UNIT & INTEGRATION TESTING

- PROGRAM DESIGN → CODING
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

Prototyping Model

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

Diagram:

- SYSTEM REQUIREMENTS (sometimes informal or incomplete)
- LIST OF REVISIONS
  - Revise prototype
- user/Customer review
  - Revisions
- TEST
- DELIVERED SYSTEM

- PROTOTYPE REQUIREMENTS
- PROTOTYPE DESIGN
- PROTOTYPE SYSTEM
2.2 Software Process Models
Operational Specification Model

- Requirements are executed (examined) and their implication evaluated early in the development process
- Functionality and the design are allowed to be merged
2.2 Software Process Models
Transformational Model

- Fewer major development steps
- Applies a series of transformations
  - Change data representation
  - Select algorithms
  - Optimize
  - Compile
- Relies on formalism
- Requires formal (to allow transformations)
2.2 Software Process Models
Phased Development: Increments and Iterations

- Shorter cycle time
- System delivered in pieces
  - enables customers to have some functionality while the rest is being developed
- Allows two systems functioning in parallel
  - the production system (release n): currently being used
  - the development system (release n+1): the next version
2.2 Software Process Models
Phased Development: Increments and Iterations (continued)

- **Incremental development**: starts with small functional subsystem and adds functionality with each new release
- **Iterative development**: starts with full system, then changes functionality of each subsystem with each new release
Phased Development: Increments and Iterations
(continued)

• Phased development is desirable for several reasons
  – Training can begin early, even though some functions are missing
  – Markets can be created early for functionality that has never before been offered
  – Frequent releases allow developers to fix unanticipated problems globally and quickly
  – The development team can focus on different areas of expertise with different releases
2.2 Software Process Models

Spiral Model

• Suggested by Barry Boehm (1988)
• Combines development activities with risk management to minimize and control risks
• The model is presented as a spiral in which each iteration is represented by a circuit around four major activities
  – Plan
  – Determine goals, alternatives and constraints
  – Evaluate alternatives and risks
  – Develop and test
2.2 Software Process Models
Spiral Model (continued)

Figure 2.10 the spiral model.
2.2 Software Process Models

Agile Methods

• Emphasis on flexibility in producing software quickly and capably

• Agile manifesto
  – Value individuals and interactions over process and tools
  – Prefer to invest time in producing working software rather than in producing comprehensive documentation
  – Focus on customer collaboration rather than contract negotiation
  – Concentrate on responding to change rather than on creating a plan and then following it
2.2 Software Process Models
Agile Methods: Examples of Agile Process

- Extreme programming (XP)
- Crystal: a collection of approaches based on the notion that every project needs a unique set of policies and conventions
- Scrum: 30-day iterations; multiple self-organizing teams; daily “scrum” coordination
- Adaptive software development (ASD)
2.2 Software Process Models
Agile Methods: Extreme Programming

• Emphasis on four characteristics of agility
  – Communication: continual interchange between customers and developers
  – Simplicity: select the simplest design or implementation
  – Courage: commitment to delivering functionality early and often
  – Feedback: loops built into the various activities during the development process
2.2 Software Process Models
Agile Methods: Twelve Facets of XP

- The planning game (customer defines value)
- Small release
- Metaphor (common vision, common names)
- Simple design
- Writing tests first
- Refactoring

- Pair programming
- Collective ownership
- Continuous integration (small increments)
- Sustainable pace (40 hours/week)
- On-site customer
- Coding standard
2.2 Software Process Models
Sidebar 2.2 When Extreme is Too Extreme?

• Extreme programming's practices are interdependent
  – A vulnerability if one of them is modified

• Requirements expressed as a set of test cases must be passed by the software
  – System passes the tests but is not what the customer is paying for

• Refactoring issue
  – Difficult to rework a system without degrading its architecture
Scrum Cartoon

Hey Pig, I was thinkin' we should open a restaurant.
I don't know. What would we call it?

How about "Ham-n-Eggs."

No thanks, I'd be committed, but you'd only be involved!

2.7 What this Chapter Means for You

• Process development involves activities, resources, and product
• Process model includes organizational, functional, behavioral and other perspectives
• A process model is useful for guiding team behavior, coordination and collaboration