A Generic Process Model
Process Flow

These slides are designed to accompany Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied
A process pattern describes a process-related problem that is encountered during software engineering work, identifies the environment in which the problem has been encountered, and suggests one or more proven solutions to the problem.

Stated in more general terms, a process pattern provides you with a template—a consistent method for describing problem solutions within the context of the software process.
Process Pattern Types

- **Stage patterns**—defines a problem associated with a framework activity for the process.
- **Task patterns**—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice.
- **Phase patterns**—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.
Process Assessment and Improvement

- **Standard CMMI Assessment Method for Process Improvement (SCAMPI)** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.

- **CMM-Based Appraisal for Internal Process Improvement (CBA IPI)** — provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment [Dun01]

- **SPICE** — The SPICE (ISO/IEC15504) standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process. [ISO08]

- **ISO 9001:2000 for Software** — a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies. [Ant06]
Prescriptive Models

- Prescriptive process models advocate an orderly approach to software engineering
  
  *That leads to a few questions …*

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?

- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?
The Waterfall Model

- Communication: project initiation, requirement gathering
- Planning: estimating, scheduling, tracking
- Modeling: analysis, design
- Construction: code, test
- Deployment: delivery, support, feedback

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
The V-Model

These slides are designed to accompany Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
The Incremental Model

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
Evolutionary Models: Prototyping
Evolutionary Models: The Spiral

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
Evolutionary Models: Concurrent

- None
- Under development
- Awaiting changes
- Under revision
- Under review
- Baselined
- Done

These slides are designed to accompany Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
Still Other Process Models

- **Component based development**—the process to apply when reuse is a development objective
- **Formal methods**—emphasizes the mathematical specification of requirements
- **AOSD**—provides a process and methodological approach for defining, specifying, designing, and constructing aspects
- **Unified Process**—a “use-case driven, architecture-centric, iterative and incremental” software process closely aligned with the Unified Modeling Language (UML)
The Unified Process (UP)

These slides are designed to accompany Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
UP Phases

Requirements
Analysis
Design
Implementation
Test
Support

Iterations
#1
#2
#n-1
#n

These slides are designed to accompany Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.
UP Work Products

### Inception phase
- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment
- Project plan, phases and iterations
- Business model, if necessary
- One or more prototypes

### Elaboration phase
- Use-case model
- Supplementary requirements, including non-functional
- Analysis model
- Software architecture
  - Description
- Executable architectural prototype
- Preliminary design model
- Revised risk list
- Project plan, including iteration plan
  - Adapted workflows
  - Milestones
- Technical work products
- Preliminary user manual

### Construction phase
- Design model
- Software components
- Integrated software increment
- Test plan and procedure
- Test cases
- Support documentation
  - User manuals
  - Installation manuals
  - Description of current increment

### Transition phase
- Delivered software increment
- Beta test reports
- General user feedback

Personal Software Process (PSP)

- **Planning.** This activity isolates requirements and develops both size and resource estimates. In addition, a defect estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.

- **High-level design.** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.

- **High-level design review.** Formal verification methods (Chapter 21) are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.

- **Development.** The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.

- **Postmortem.** Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.
Team Software Process (TSP)

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
  - The Capability Maturity Model (CMM), a measure of the effectiveness of a software process, is discussed in Chapter 30.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills.