Introduction to Software Engineering Processes

SWENET SEP1 Module

Developed with support from the National Science Foundation

Overview

- Software Development Models
- Software Processes
- CMM-type Processes
- Agile Processes
- Process Exercise
- References

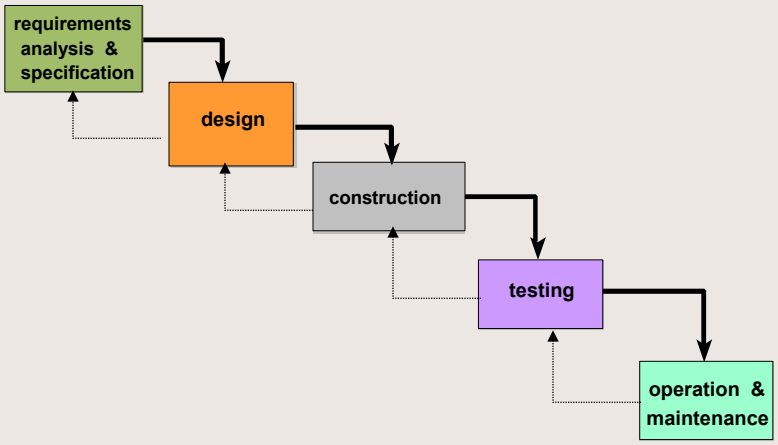
Software Development Models

- Over the years, various models have emerged to support the development of software products.
- The models address the following life-cycle activities:
  - requirements analysis and specification
  - design
  - construction
  - testing
  - operation and maintenance

Waterfall Development Model - 1

- The Waterfall life cycle model emphasizes that software is developed in sequential phases (e.g., analysis, design, code, etc.) with established milestones, documents, and reviews at the end of each phase.
- In the pure waterfall there is no overlap of phases; however, in refinement of the model, “feedback loops” are introduced to allow return to previous phases (e.g., design analysis leads back to modification of the requirements)

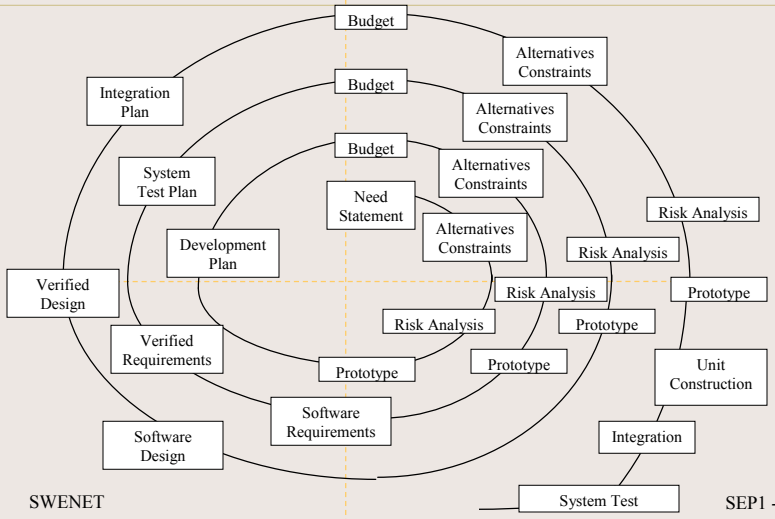
Waterfall Development Model - 2



Spiral Development Model -1

- The Spiral Lifecycle (diagrammed on the next slide as a sequence of cycles) combines elements of the waterfall lifecycle model, along with an emphasis on the use of risk management techniques.
- Each cycle includes the following phases: Determine goals, alternatives,constraints; risk analysis; prototype development; product development and verification; and planning for next phase.

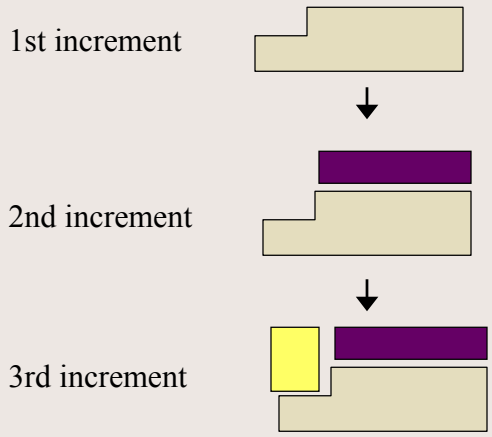
Spiral Development Model - 2



Incremental Development Model - 1

- For the Incremental Model an overall architecture of the total system is developed first and the detailed increments and releases are planned.
- Each increment has its own complete lifecycle.
- The increments may be built serially or in parallel depending on the nature of the dependencies among releases and on availability of resources.
- Each increment adds additional or improved functionality to the system.

Incremental Development Model - 2



Software Processes - 1

- A process is a series of steps involving activities, constraints and resources that produce an intended output of some kind.
- A software process (or a software engineering process) guides and supports the development of a software product.
- In the last decade there has been a great of deal of resources devoted to the definition, implementation, and improvement of software development processes.
- The list of references at the end of this presentation refer to papers that provide background and discussion for further study of software process features and issues.

Software Processes - 2

- A “defined” software process would have a documented description of its features which would typically include the following:
  - scripts that define the process steps
  - standards and procedures for carrying our process steps
  - forms and templates for collecting process data and documenting process outcomes

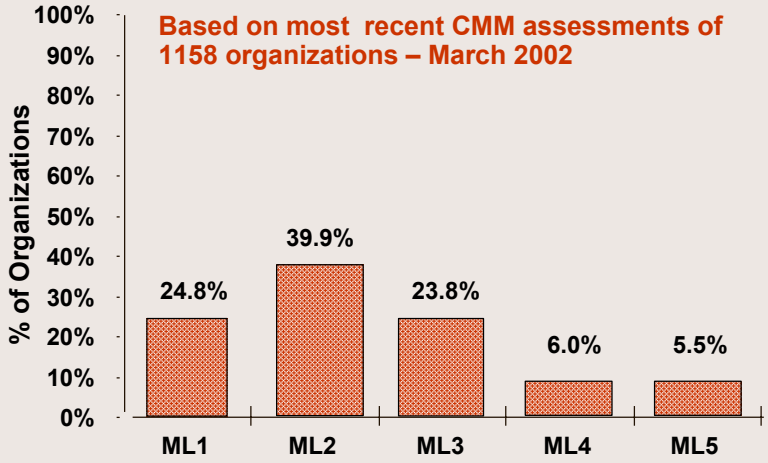
A Process Framework

- In the late 1980s, the Software Engineering Institute (SEI), Carnegie Mellon University, developed the SW-CMM (Software Capability Maturity Model) to help organizations build effective software engineering processes [6]. It has been widely adopted in industry, primarily by large software development organizations.
- The next slide describes the five-level model encompassing good engineering and management practices.
- Many software development organizations have been assessed using the SW-CMM framework. Results of such assessments are shown in a later slide.

SW-CMM Level Description

Level	Focus	Key Process Areas
5 Optimizing	Continuous process improvement	Defect prevention Technology change management Process change management
4 Managed	Product and process quality	Quantitative process management Software quality management
3 Defined	Engineering process	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
2 Repeatable	Project management	Requirements management Software project planning Software project tracking Software subcontract management Software quality assurance Software configuration management

SW-CMM Assessment

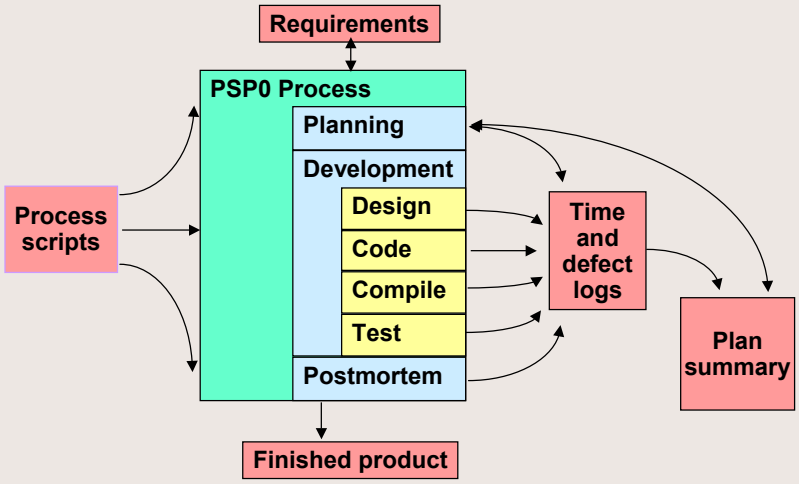


A Simple Software Process

- The Personal Software Process (PSP[SM]) is software process developed at the SEI to address some of the SW-CMM practices at the level of the individual programmer [3].
- PSP0 is the simplest version of PSP.
  - It has three phases: planning, development and postmortem.
  - The development phase has four sub-phases: design, code, compile, and test.
  - There are process scripts that describe what happens in each phase.
  - There are logs and a summary sheet for collecting and analyzing data.
  - The next two slides provide a graphic that depicts the PSP0 process elements and an example of a process script.
- What kind of software development model does the PSP use?

[SM]Perosnal Software Process and PSP are service marks of Carnegie Mellon University.

The PSP0 Process Flow



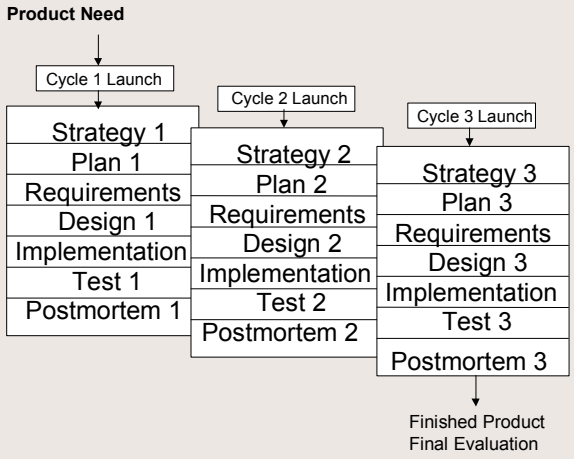
PSP0 Process Script

	Purpose:	To guide you in developing module-level programs.
	Inputs Required	Problem description PSP0 project plan summary form Time and defect recording logs Defect type standard Stop watch (optional)
1	Planning	- Produce or obtain a requirements statement. - Estimate the required development time. - Enter the plan data in the project plan summary form. - Complete the time log.
2	Development	- Design the program. - Implement the design. - Compile the program and fix and log all defects found. - Test the program and fix and log all defects found. - Complete the time recording log.
3	Postmortem	Complete the project plan summary form with actual time, defect, and size data.
	Exit Criteria	- A thoroughly tested program - Completed project plan summary with estimated and actual data - Completed defect and time logs

A Team Software Process

- The SEI also developed a Team Software Process (TSP) that includes most of the key process areas of the SW-CMM [7].
- Software is developed in multiple cycles with each cycle made of multiple phases.
- An academic version of the TSP consist of three cycles with teams consisting of five members, each with a well defined role: a team leader, a development manager, a planning manager, a quality/process manager and a support manager.

TSP Structure



Agile Processes

- Some have criticized highly-structured processes (such as those based on a CMM framework) as unresponsive to change during a development cycle (e.g., change in requirements or change in technology).
- So-called “agile methods” have been developed to address this criticism and reduce the cost of change throughout a project.
- Agile developers advocate the following [5]:
  - individual and interactions over processes and tools
  - working software over comprehensive software documentation
  - customer collaboration over contract negotiation
  - responding to change over following a plan
- One of the most widely practiced agile methodologies is Extreme Programming (XP).

Extreme Programming

- XP is designed for work in small to medium teams (less than 10 members), building software with vague or rapidly changing requirements.
- The XP life cycle has four basic activities [6]:
  - continual communication with the customer and within the team
  - simplicity, achieved by a constant focus on minimalist solutions
  - rapid feedback through unit and functional testing
  - emphasis on dealing with problems proactively

XP Practices

- XP typically consists of 12 practices [6]. Here are a few:
  - *Small Releases*: Put s simple system into production quickly. Release new version on a very short (2 week) cycle.
  - *Simple Design*: Design as simply as possible at any given moment.
  - *Testing*: Developers continually write unit tests that must run flawlessly.
  - *Pair Programming*: All production code is written by two programmers at one machine.
  - *40-hour weeks*: Work no more than 40 hours per week whenever possible.
  - *On-site Customer*: Have a customer on the team full-time to answer questions.
  - *Coding Standards*: Have rules that emphasize communication throughout the code.

Process Exercise

- The exercise booklet for this module includes reading and discussion about software processes.
- The exercise also includes a problem where you define a process for carrying out some familiar activity.
- After you complete the exercise, we will discuss it in class.

References

1. Bourque P. and Dupuis R., eds. *Guide to the Software Engineering Body of Knowledge*, IEEE CS Press, Los Alamitos, Calif., 2001. (http://www.swebok.org/)
2. Brooks, F. P., *The Mythical Man Month*, Chapter 2: "The Mythical Man Month", pp. 13-26, Addison-Wesley, 1999.
3. Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A. "Introducing the Personal Software Process: Three Industry Case Studies," *Computer*, pp. 24-31, May 1997.
4. Fleming, R., "A Fresh Perspective on Old Problems", *IEEE Software*, pp 106-113, January 1999.
5. Highsmith, J. and Cockburn, A., "Agile Development: The Business of Innovation", pp. 120-122, *Computer*, September 2001.
6. Paulk, M., "Extreme Programming from a CMM Perspective", *IEEE Software*, pp. 19-26, November 2001.
7. Webb, D. and Humphrey, W. S., "Using the TSP on the TaskView Project", *CrossTalk, Journal of Defense Software Engineering*, pp. 3-10, February 1999. (http://www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1999/02/webb.asp)