Notes Jan 23, 2019

UML Project

* Your goal is to demonstrate that you can use UML to illustrate important information about a system. You may choose a system that you already know about or one that you make up. Turn in the following (in the order shown):
  + No more than one page describing your system. Don’t try to duplicate the information in your UML diagrams completely. Just tell me what your system does at a fairly high level. After reading this description I should be able to understand the basic purpose of the remaining UML diagrams. Context of this should be as though you’re proposing to build this system.
  + A narrative section describing the requirements of the system.
  + A use-case diagram (comprised of multiple use cases) showing a representative sample of a functionality of the system. However, you don’t have to nuclide every single. Function/requirement in the models.
  + Activity diagrams that add detail to at least two of your cases. These can be standard activity or swim lane diagrams.
  + At least one class diagram.
* Your diagrams should conform to UML standards, don’t include elements that are non-standard because that may obscure.

Chapter 3

* Software Process Structure
  + Slide Set to accompany
  + Software Engineering: A Practitioner’s Approach, 8/e
    - By Roger S. Pressman and Bruce R. Maxim
    - Slides copyright 1996, 2001, 2005, 2009, 2014 by Roger S. Pressman
  + For non-profit educational use only
    - May be reproduced ONLY for student use at the university level when used in conjunction.

A Generic Process Model

* Software Process
  + Process Framework
    - Umbrella Activities
      * Framework activity #1
        + Software

Process Flow

* 🡪 Communication 🡪 Planning 🡪 Modeling 🡪 Construction 🡪 Deployment 🡪
  + (a) Linear process flow
* 🡪 Communication 🡪 Planning 🡪 Modeling 🡪 Construction 🡪 Deployment 🡪

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 tasks to be accomplished
  + A list of the work products to be produced

Process Patterns

* 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 [Amb98] – 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

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 [Duno1]
* SPICE – The SPICE (ISO/IEC15504)

Chapter 4

* Process models
  + Slide Set to accompany
  + Software Engineering: A Practitioner’s Approach, 8/e
    - By Roger S. Pressman and Bruce R. Maxim

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.

The Waterfall Model

* 🡪 Communication (project initiation requirements gathering) 🡪 Planning (estimating, scheduling, tracking) 🡪 Modeling (analysis, design) 🡪 Construction (code, test) 🡪 Deployment (delivery, support, feedback) 🡪

The V – Model

* Requirements modeling
  + Architectural Design
    - Component Design
      * Code Generation
        + Executable Software
      * Unit Testing
    - Integration Testing
  + System Testing
* Acceptance Testing

The Incremental Model

* (1) Communication
* (2) Planning
* (3) Modeling (analysis, design)
* (4) Construction (code, test)
* (5) Deployment (delivery, feedback)
* Increment # 1

Evolutionary Models: Prototyping

* Communication 🡪 Quick Plan 🡪 Modeling Quick design 🡪

Evolutionary Models: The Spiral

* Communication 🡪 Planning (estimation scheduling risk analysis)

Evolutionary Models: Concurrent

* Inactive 🡪

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.

The Unified Process (UP)

* Inception
  + Communication, Planning
* Elaboration
  + Planning, Modeling
* Construction
* Transition
  + Construction, Deployment
* Production
  + Software Increment (Release)

UP Phases

* Inception Elaboration Construction

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.

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. An external

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 (See Ch 30) behavior normal and expected.
* Provide improvement guidance.