

3. The Software Process

1. Introduction
2. Requirements Capture
3. Software Processes
 - 3.1 Some Background
 - 3.2 Unified Process
 - 3.3 Extreme Programming
 - 3.4 Open Source Programming
4. Analysis
5. Design
6. Quality Assurance and Implementation

497-3-1

3.1 Background about Processes

- Process:
 - a partially ordered set of steps
- Process model:
 - a set of similar processes
- Meta-process
 - Capability Maturity Model (CMM)
- Process principles
 - Quality management
 - Risk management
 - Role-based development
 - Milestones
 - Checkable
 - Low overhead
 - Appropriate to task and team

497-3-2

Processes

- Kinds of clients:
 - in-house
 - contract
 - off-the-shelf software (or: shrink-wrapped)
- Size of project / team
 - influences amount of planning and management
 - influences amount of paperwork
- Process needs to be appropriate for:
 - the product
 - the people
 - the tools

497-3-3

Usual Process Phases

- Software development lifecycle
 - Requirements / Analysis / Specification
 - Design
 - Implementation / Testing
 - Maintenance
- Software use lifecycle
 - Need identification
 - Development
 - Operation
 - Retiring

497-3-4

Products of Development

Activities

Requirements	informal text, understandable by user
Analysis	conceptual model, business model
Design	design model (dependent on programming language, operating system, libraries)
Implementation	program
Testing	fault-free program, test report
Maintenance	updated versions of program

497-3-5

Requirements Capture

- Determine what user/client wants
- Agree on a contract
- What problem is the program supposed to solve?
- Difficulties:
 - Users do not know what they want
 - Users may ask for impossible things
 - Requirements are usually vague, incomplete, contradictory
- Communication between user and developer

497-3-6

Analysis

- Understand the problem domain
 - Business modeling
 - Workflow modeling
- Look for unknown terms and concepts
 - Technical concepts of the user domain
 - Background knowledge needed to understand the problem
- Understand the tasks that users want to perform
- Identify conceptual classes and their relationships
 - Just enough modeling to express the tasks of the program

497-3-7

Design

- Take the analysis model and refine it into something that can be implemented
- Consider concerns of the programming language
 - Multiple inheritance?
 - Interfaces?
 - Built-in data types?
- Consider reuse
 - Standard libraries
 - Commercial software
- Other technical concerns
 - GUI
 - Database
 - Performance
- Add enough detail so that implementation will be easy

497-3-8

Implementation

- Working code!

497-3-9

Testing

- Find all kinds of faults and remove them
- Alpha-testing: in-house
- Beta-testing: by users
- Unit test: by developer
- System test: by opposing tester
- Faults introduced in:
 - Requirements
 - Analysis
 - Design
 - Implementation
- Different testing techniques find different faults

497-3-10

Maintenance

- All development after release
- Upgrades (“perfective”)
 - New features
- Fixes (“corrective”)
 - Problems that should have been solved before
- Reasons:
 - Users change their minds
 - Environment changes
 - Better technologies become available

497-3-11

Some Process Models

- Waterfall
 - sequential with little feedback
- Incremental
 - waterfall in a loop
- Synchronize and Stabilize
 - concurrent
- Spiral
 - risk management and verification
- Unified Software Development Process
 - complex, object-oriented, adaptable
- Extreme Programming
 - light-weight, flexible, small
- Open Source
 - widely distributed development

497-3-12