Software Life Cycle and Models

Rajkumar Buyya
Grid Computing and Distributed Systems Lab
Dept. of Computer Science and Software Engineering
University of Melbourne, Australia
http://www.buyya.com

Software Process

- Software Process defines the way to produce software. It includes
  - Software life-cycle model
  - Tools to use
  - Individuals building software
- Software life-cycle model defines how different *phases* of the life cycle are managed.

Phases of Software Life-cycle

- Requirements
- Specification (Analysis)
- Design
- Implementation
- Integration
- Maintenance
- Retirement

Requirements

- Assumption
  - The software being considered is considered economically justifiable.
- Concept exploration
  - Determine what the client needs, *not* what the client wants
- Document - Requirements Document

Specification (Analysis) Phase

- From the customer requirements identify *what* to build.
- Specifications must not be
  - Ambiguous
  - Incomplete
  - Contradictory
- Document – Specification Document

Design Phase

- From the specification identify *how* to build.
- Design involves two steps
  - Architectural Design – Identify modules
  - Detailed Design – Design each modules
- Document – Architecture Document, Design Document

Implementation Phase

- Implement the detailed design in code.
- Developer testing
  - Unit testing
  - Module testing
- Document – Commented source code

Integration Phase

- Combine the modules and test the product as a whole.
- Testing includes
  - Product testing
  - Acceptance testing
- Document – Test cases and test results

Maintenance Phase

- Any changes after the customer accepts the system.
- Maintenance phase is the most expensive
  - Lack of documentation
  - Regression testing
- Document – Documented Changes, Regression test cases

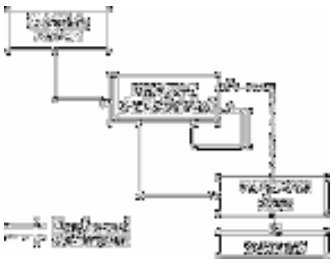
Retirement Phase

- Good software is maintained
- Sometimes software is rewritten from scratch
  - Software is now un-maintainable because
    - A drastic change in design has occurred
    - The product must be implemented on a totally new
    - hardware/operating system
    - Documentation is missing or inaccurate
    - Hardware is to be changed—it may be cheaper to rewrite
    - the software from scratch than to modify it
- True retirement is a rare event

Life-Cycle Models

- Build-and-fix model
- Waterfall model
- Rapid prototyping model
- Incremental model
- Extreme programming
- Synchronize-and-stabilize model
- Spiral model
- Object-oriented life-cycle models
- Comparison of life-cycle models

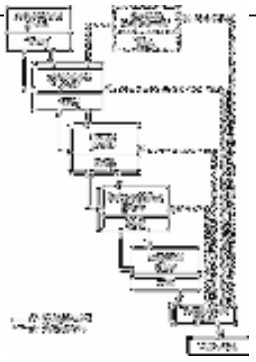
Build and Fix Model



Notes

- Most software is developed using build-and-fix model. Basically there is no model.
  - No specifications
  - No design
- This model is completely unsatisfactory and should not be adopted.
- Need life-cycle model
  - “Game plan”
  - Phases
  - Milestones

Waterfall Model



Notes

- Output from one phase is fed as input to the next phase.
- One phase is completed, documented and signed-off before the next phase begins.
- Advantages
  - Each phase is well documented.
  - Maintenance easier.
- Disadvantages
  - If there is a mismatch between what the client wanted and was is built this will not be known till the product is delivered.

Rapid Prototyping Model

Notes

- A prototype of the product is build rapidly and shown to the client before the product is completely built.
- Advantages :
  - Any mismatches between requirement and the product can be found early.
- Disadvantages :
  - Sometimes the prototype ends up being the final product which results in quality, maintenance problems.

Summary

- Software Engineering is an important discipline due to various aspects.
- Analysis and Design are two very important phases in the software development lifecycle.

Reference

- Stephen Schach, *Classical and Object-Oriented Software Engineering with UML and Java,* Chapter 3, McGraw-Hill, New York, USA.
  - http://www.mhhe.com/engcs/compsci/schach5/samplech.mhtml
- Any other book on software engineering is also fine!