Software Engineering Notes

Patrick Lindsay

January 17, 2013

Contents

Part I

Notes

Chapter 1

January 15, 2012

1.1 Overview

Software Engineering - The application of sound engineering practices to software creation and maintenance.

1.1.1 Software development Life Cycle (Traditional Approach)

- Requirements Phase
- Analysis or Specification Phase
- Design Phase
- Implementation/Integration Phase
- Maintenance Phase
- Retirement

Requirements Phase

- Determining the NEEDS and WANTS of the client or customer.
- Determining the constraints of the system.

Analysis or Specification Phase

- After analyzing the requirements, construct a *specification document* which explicitly describes what the product is to do, and the constraints under which it must operate.
- This includes the description of the input, output, actions, and UI.

- The specification document can be used as part of a contract with the client.

Problems with the Spec Document

1. Ambiguity - one sentence may have more than one interpretation.
2. Incompleteness - relevant fact or requirement is left out.
3. Contradiction - two places in the spec document are in conflict.

Design Phase

- Construct an *Architectural Design*.
- Construct a *Detailed Design*.
- Test for *traceability*.

1. Architectural Design - Description of the product in terms of modules.
2. Detailed Design - Description of each module.
3. Traceability - each part of the design can be traced to a statement in the specification document.

Implementation Phase

- Code each module from the detailed design.
- Programmer tests his/her own code separately.
- Modules are combined and tested by developers.
- Product is tested by SQA group. This is called product testing.
- Project is given to the client for acceptance testing.

Maintenance Phase

- Corrective Maintenance - bug squashing
- Enhancement Maintenance - Updates
  - Perfective - client makes new demands
  - Adaptive - changes in the environment of the product requires changes in the software.
- Perform regression testing - insuring that changes have not affected already working functionality.

Retirement Phase

- Determining if desired changes are too costly.
- Determining if a product is obsolete.

1.1.2 Four Components of the Software Engineering Enterprise

The four P's

1. Process
2. Project
3. People
4. Product

Process

- The process is sometimes called the life-cycle model or development sequence.
  - Waterfall
  - Spiral
  - Incremental Build
- Makes use of several process frameworks.
  - Personal Software Process (PSP)
  - Team Software Process (TSP)
  - Capability Maturity Model (CMM)
- Documentation Standards
  - IEEE
  - ANSI

Project

- The set of activities needed to produce the required product.
- Project management is extremely important.
- Many projects are not about developing new products, but maintaining already existing *legacy* systems.

People

- Team Organization
- Team Management
- Relationship with customer or client
- Relationship with end users
- Communication with upper management

Product

Includes

- Requirement Specification Document
- Design Document
- Source Code
- Executable
- User Manuals

Chapter 2

January 17, 2013

2.1 Traditional Software Engineering Process

2.1.1 Historical Influences

- Structured programming (Edsger Dijkstra's letter calling "GOTOs" harmful) uses sequence control, iteration, invoking functions
- Object Oriented paradigm: the use of objects with data and functionality which can represent real-world entities.

Note:
Silver Bullet ca. 1980s
Likened the software crisis to a werewolf. Object Oriented Paradigm was the Silver Bullet.
It did not work as expected.

- Design Patterns: stock of reusable design elements (templates)

2.1.2 Component Reuse

A component as defined by Meyer is " a program element satisfying:"

1. The element may be used by other program elements. (Clients)
2. The clients and their authors do not need to be known to the element's author

2.1.3 Key Expectations of Software Engineering

1. Decide in advance what the specific quality measures are to be for the project and product.
   Predetermine quantitative quality goals.

2. Gather data on all projects to form a basis for estimating future projects.

3. All requirements, designs, code and test materials should be freely and easily available to all members of the team.
   Source code should always be available to all team members in an easily accessible and interpreted way.
   Git, Mercurial, etc.

4. A process should be followed by all team members. *Uniformity*

   (a) Design only against requirements.

   (b) Program only against design.

   (c) Test only against requirements and design.
   ALWAYS FOLLOW THE RECIPE!

5. Measure and achieve quality goals.

2.2 Methods

Be able to draw and discuss these.

2.2.1 Waterfall Method

SEE DIAGRAM 52.9 ON PG 53.

- First described by William*(?)* Royce in 1970.
- No phase is complete until documentation for that phase has been completed and approved by the SQA group.
  Very orderly; heavy on documentation.
- Has been used with great success on a variety of products.
- Feedback loops permits modifications to be made to the previous phase.

Advantages

1. Enforced disciplined approach
2. Requirement that documentation be provided at each phase.
3. All products of the phase must be checked by SQA.
4. Inherent aspect of each phase is testing.

Disadvantages

- The resulting specification document may not be able to be understood by the client.
- It can lead to the construction of product that does not meet the client's needs.

2.2.2 Rapid Prototype Model

See diagram on pg 55. Construction of a functional subset of the desired product in order to allow the client and the developer to interact.
Keyword is rapid. This is a thrown-together, proof-of-concept type project; a mock-up.

Advantages

- The process is linear and possibly faster than the Waterfall Model
- Increases interaction between client and developer.

Disadvantages

- Client may inaccurately think the product is almost complete when viewing the prototype.
- Developer may attempt to use the prototype as part of the final product.

2.2.3 Waterfall-Rapid Prototype Hybrid

May form a hybrid model using the rapid prototype as the first phase in the Waterfall Model in order to increase interaction but allow for feedback loops within the development of the product.

2.2.4 Incremental Model

Software is implemented, integrated, and tested as a series of incremental builds.
Code pieces providing specific functions.

Advantages

1. Results in builds which can be developed in weeks, not months or years.
2. End user need not learn the entire product at one time.
3. Client need not pay for the entire product at one time.
4. Developer gets paid earlier.
   (At each build delivery)

5. Open-ended design makes maintenance easier.

6. Easier to make changes during development.

Disadvantages

1. Each new build must fit in without destroying existing builds.
   Regression Testing

2. Requires more careful to design to make it open to additions.

3. Can degenerate to a build and fix product if broken into too few builds.

2.2.5 Spiral Model

See Figure 2.12 on pg 63 and Figure 2.13 on pg 65.

- A Waterfall Model with each phase preceded by risk analysis in an attempt to control or resolve risk.
- Each phase is 360.
- The measure of the radius is the cumulative cost to date.
- The measure of the angle is the progress measure.
  Each phase is 360.
  Requires a very experienced engineer.

Advantages

1. The emphasis on alternatives and constraints supports the reuse of existing software.

2. The incorporation of software quality as a specific objective.

3. Answers the question of how much testing should be performed in terms of risks.

4. Maintenance is simply another cycle of the spiral, the same as development.

Disadvantages

1. Intended exclusively for internal development.
   Client and developer are members of the same organization.

2. *Applicable only to large-scale projects.*

3. Must have developers who are skilled at pinpointing the possible risks.

2.3 Agile Methods

2.3.1 General

According to the Agile Manifesto, they value:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responsiveness to change over following a plan.
  YOU DON'T ALWAYS HAVE TO FOLLOW THE RECIPE!

Traits

- Highly iterative
- Pair programming with a focus on teamwork and ego-less programming.
  THIS IS MANDATORY.
- Early and planned testing.
- Story cards *Similar to storyboards in movies.*
- Refactoring *Turning working code into better code.*
- Feedback

Principles Behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive disadvantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.