Notes Jan 16 2019

Chapter 1

1. Instructions

What is Software?

* Software is developed or engineered, it is not manufactured in the classical sense.
* Software doesn’t “wear out.”
* Although the industry is moving toward component-based construction, most software continues to be custom built

Wear vs. Deterioration

* Failure Rate vs Time
  + Increased failure rate due to side effects
  + Change
  + Actual curve
  + Idealized curve

Software Applications

* System Software
* Application Software
* Engineering/Scientific Software

Legacy Software

* Software must be adapted to meet the needs of new computing environments or technology
* Software must be enhanced to implement new business requirements
* Software must be extended to make it interoperable with other more modern systems or databases.
* Software must be rearchitected to make it viable within a network environment.

Mobile Apps

* Reside on mobile platforms such as cell phones or tablets
* Contain user interfaces that take both device characteristics and location attributes
* Often provide access to a combination of web-based resources and local device processing and storage capabilities
* Provide persistent storage capabilities within the platform
* A mobile web application allows a mobile device to access to web-based content using aa browser designed to accommodate the strengths and weaknesses of the mobile platform
* A mobile app can gain direct access to the hardware found on the device to provide local processing and storage capabilities.
* Ad time passes these differences will become blurred.

Cloud Computing

* Servers, Desktops, Laptops, Phones, Tablets
  + Application
    - Monitoring, Content, Collaboration, Communication,
  + Platform
    - Object Storage, Identity, Runtime, Queue, Database
  + Infrastructure
    - Compute, Black Storage, Network

Cloud Computing

* Cloud computing provides distributed data storage and processing resources to networked computing devices
* Computing resources reside outside the cloud and have access to a variety of resources inside the cloud
* Cloud computing requires developing an architecture

Product Line Software

* Product line software is a set of software-intensive systems that share a common set of features and satisfy the needs of a particular market.
* These software products are developed using the

Characteristics of WebApps – II

* Data driven. The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.
* Content sensitive. The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
* Continuous evolution. Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.
* Immediacy. Although immediacy – the compelling need to get software to market quickly – is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.
* Security. Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.
* Aesthetics. An undeniable part of the appeal of a WebApp is its look and feel.

Chapter 2

* Software Engineering

Software Engineering

* Some Realities:
  + A concerted effort should be made to understand the problem before a software solution is developed
  + Design becomes a pivotal activity
  + Software should exhibit high quality
  + Software should be maintainable
* The seminal definition:
  + [Software engineering is] the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines.

A Process Framework

* Process Framework
  + Framework Activities

Framework Activities

* Communication
* Planning
* Modeling
  + Analysis of requirements
  + Design
* Construction
* Deployment

Umbrella Activities

* Software project tracking and control

Adapting a Process Model

* The overall flow of activities, actions, and tasks and the interdependencies among them
* The degree to which actions and tasks are defined within each framework activity
* The degree to which work products are identified and required
* The manner which quality assurance activities are applied
* The manner in which project tracking and control activities

The Essence of Practice

* Polya suggests:
  + Understand the problem (communication and analysis)
  + Plan a solution (modeling and software design)
  + Carry out the plan (code generation).
  + Examine the result for accuracy (testing and quality assurance)

Understand the Problem

* Who has a stake in the solution to the problem? That is, who are the stakeholders?
* What are the unknowns? What data

Plan the Solution

* Have you seen similar problems before? Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required.
* Has a similar problem been solved? If so, are elements of the solution reusable?
* Can subproblems be defined?

Carry Out the Plan

* Does the solution conform to the plan? Is source code traceable to the design model?

Examine the Result

* Is it possible to test each component part of the solution? Has a reasonable testing strategy.

Hooker’s General Principles

* The Reason It All Exists
* KISS (Keep It Simple, Stupid!)
* Maintain the Vision
* What You Produce, Others Will Consume
* Be Open to the Future
* Plan Ahead for Reuse
* Think!

Software Myths

* Affects managers, customers (and other non-technical stakeholders)
* If we get behind, we’ll just add more programmers…
* If we outsource the project to a 3rd party, just relax and wait…
* We have a high-level idea of what they want, we can start coding now and fill in details later…
* Once it compiles and passes unit test, we’re done…
* Documentation just slows us down…

Software – A Note About Safety