Ch1

- Software is a set of items or objects that form a configuration that includes ?

> instructions; data structure; documents

- Software types

> - System Softwares
> - Application Software
> - Engineering/Scientific Software
> - Embedded Software
> - Product-line Software
> - Web-App
> - AI Software

- Difference between software and hardware

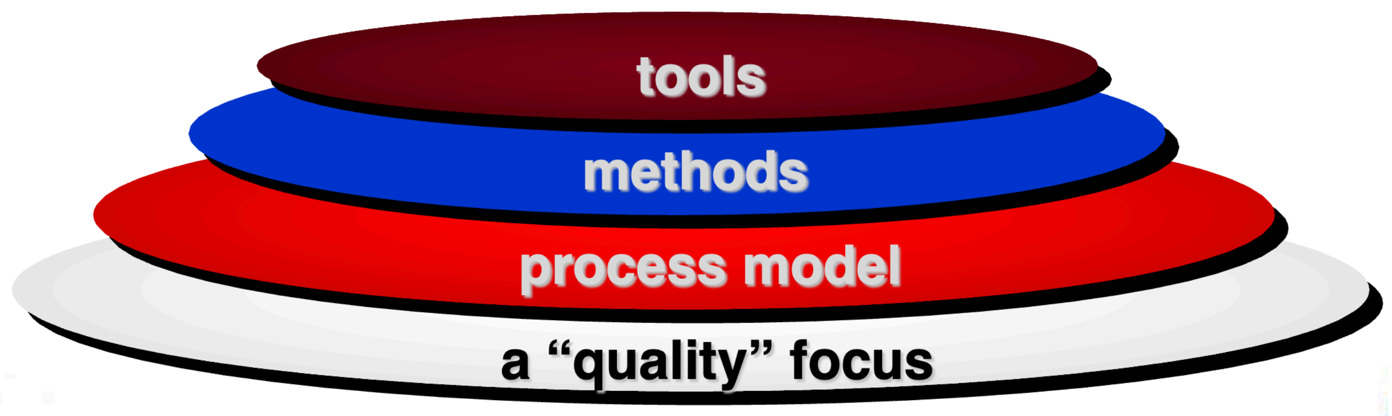
> - Software is developed or engineered. Not manufactured in the classical sense
> - Software doesn't wear out, but it does deteriorate
> - Software continues to be custom built

- Why must software change?

> - 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 **re-architected** to make it viable within a network environment.

Ch2

- 🌟 SE Layers



- 🌟🌟 5 Generic Process Framework

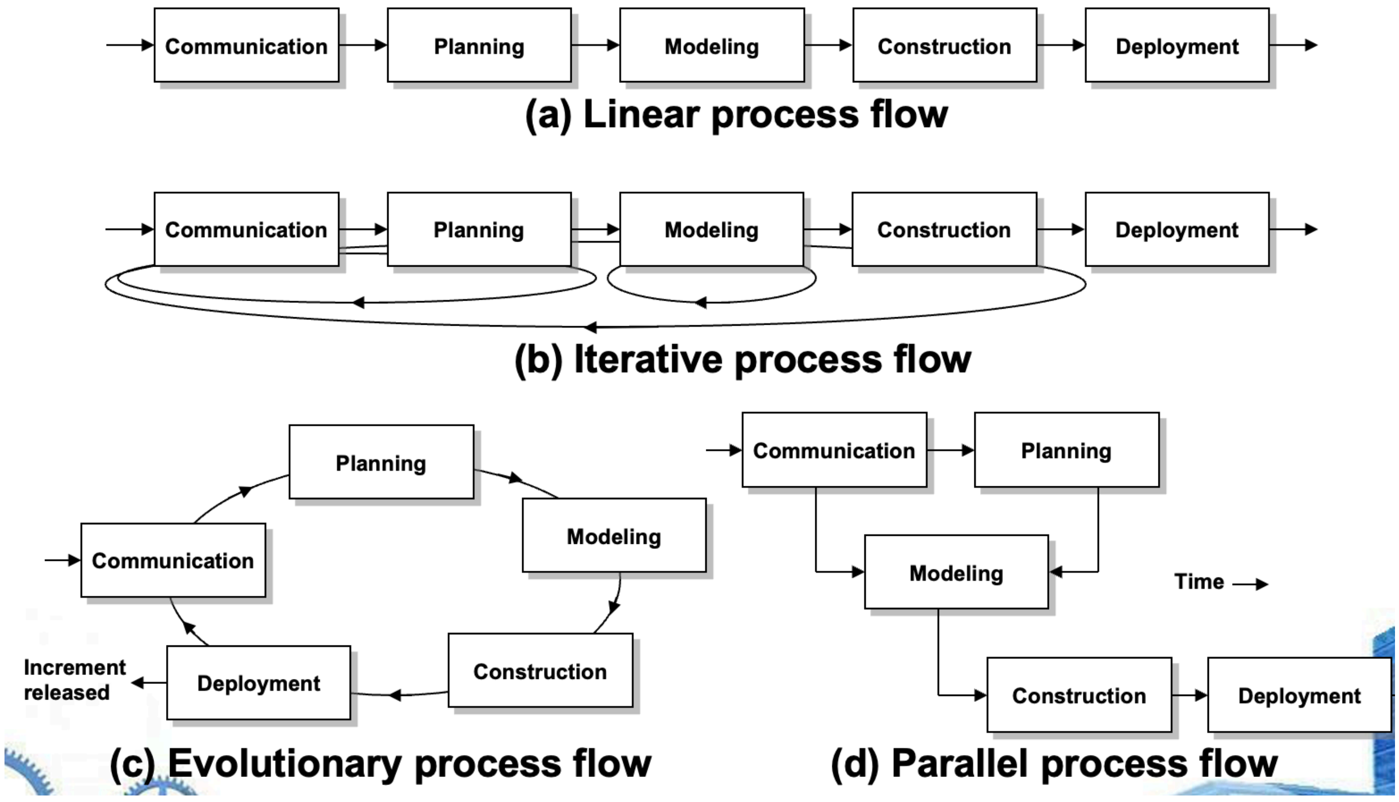
> - Communication (与客户合作，获取需求)
> - Planning (构建目标，描述难点、需求，定义工作计划)
> - Modeling (构建模型方便理解软件需求和设计)
> - Construction (软件开发和测试)

- Deployment (软件交付和用户评估反馈)

- 7 General Principles
  - The reason it all exists : Provide value to users
  - KISS : Keep it simple, stupid
  - Maintain the Vision
  - What you produce, others will consume
  - Be open to the future
  - Plan ahead for reuse
  - Think

Ch3

- ⭐ 4 Process Flow



- Process Patterns
  - Pattern name
  - Intent
  - Type
  - Initial context
  - Solution
  - Resulting context
  - Related patterns
  - Known examples

- Process Assessments

  SCAMPI, CBA IPI, SPICE, **ISO 9001:2000 for Software**

- CMMI

> Capability Maturity Model Integration, 6 levels

Ch4

- 🌟 Phases of Unified Process

> - Inception Phase
> - Elaboration Phase
> - Construction Phase
> - Transition Phase
> - Product Phase

Ch5

- Why Agile

> - Effective Response to change/communication
> - Driven by customer's requirement
> - Self-organization
> - Rapid incremental delivery

- 🌟 4 framework activities found in the Extreme Programming (XP) process model

> - Planning: 创造用户故事-分配成本-分组到一个可交付增量-用项目速度来评估
> - Design: Follow KIS, Encourage use of CRC cards, 创建尖峰解决方案, 鼓励重构
> - Coding: 编码开始前对故事进行单元测试, 鼓励结队编程
> - Testing: All unit tests are executed daily, 用户定义的验收测试

Ch8

- 🌟 8 Requirements Engineering Tasks

> - Inception: 对项目建立基本的理解(Context-Free Questions)
> - Elicitation: 询问客户的需求(Normal Expected Exciting Req/ Non-Functional Req)
> - Elaboration: 建立需求模型
> - Negotiation: 参与各方均能达到一定的满意度，实现双赢。
> - Monitoring
> - Specification: 一个规格说明可以是一份写好的文档、一套图形化的模型、一个形式化的数学模型、一组使用场景、一个原型或上述各项的任意组合。
> - Validation: 在确认这一步对需求工程的工作产品进行质量评估。Consistency / Omissions / Ambiguity
> - Management: 需求管理是用于帮助项目组在项目进展中标识、控制和跟踪需求以及需求变更的一组活动。

- 🌟 4 Analysis Models

> - Scenario-based elements: use-case diagram/activity diagram/swim lane diagram
> - Behavioral elements: state diagram

> - Flow-oriented elements: data flow diagram
> - Class-based elements: class diagram/crc models

Ch12

- 🌟 4 Designs

> - Data/Class
> - Architectural
> - Interface
> - Component

- 5 Design Model Elements

> - Data
> - Architectural
> - Interface
> - Component
> - Deployment

Ch13

- 🌟 4 Architectural Genre/Style

> - Data-Centered
> - Data Flow
> - Call and Return
> - Layered

- 3 Architectural Patterns

> - Concurrency
> - Persistence
> - Distribution

Ch14

- 🌟 7 Basic Design Principles

> - Open-Closed Principle: 开放扩展，关闭修改
> - Liskov Substitution Principle: 子类可被父类替换
> - Dependency Inversion Principles: 依赖于抽象
> - Interface Segregation Principle: Client-specific Interface
> - Release Reuse Equivalency Principle: 发布的粒度是重用的粒度
> - Common Closure Principle: 一起变的类在一起
> - Common Reuse Principle: 不一起重用的类不放一起

- Component Design for webapps

> - Content Design
> - Functional Design

- Component Based Development

> - OMG/CORBA
> - Microsoft COM
> - Sun JavaBeans

- CBSE Process (Component Based Software Engineering)

> - Qualification
> - Adaptation
> - Composition
> - Update

Ch15

- 🌟 Three Gold Rules

> - Place the user in control
> - Reduce the user's memory load
> - Make the interface consistent

- 🌟 4 Interface Analysis and Design Models

> - User Model
> - Design Model
> - Mental Model/System Perception
> - Implementation Model

Ch16

- Three-part rule

> Context, problem, solution

- WebApp Patterns

> - Information Architecture
> - Navigation patterns
> - Interaction patterns
> - Presentation patterns
> - Functional patterns

Ch17

- Two Basic Approaches

> Artistic ideal & Engineering ideal

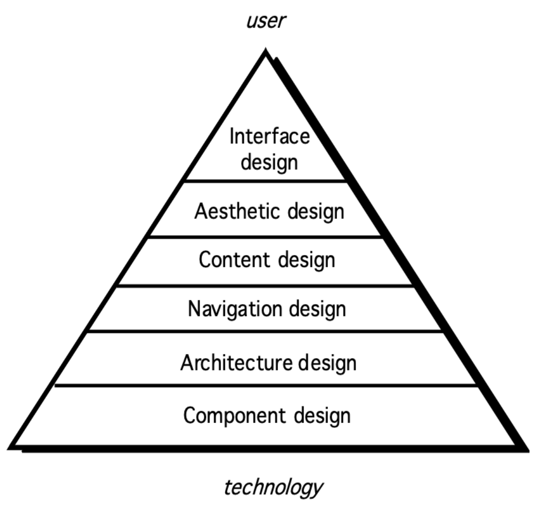
- WebApp Design Quality

> Security/Availability/Scalability/Time to market

- 🌟 6 Design Goals

> Simpilicity/Consistency/Robustness/Navigability/Visual Appeal/Compatibility

- 🌟 WebApp Design Pyramid



> 🌟 Aesthetic:
>
> - Don’t be afraid of white space.
> - Emphasize content.
> - Organize layout elements from top-left to bottom right.
> - Group navigation, content, and function geographically within the page.
> - Don’t extend your real estate with the scrolling bar.
> - Consider resolution and browser window size when designing layout.
>
> Architecture:MVC
>
> - model :contains all application specific content
> - view :contains all interface specific functions
> - control:manages access to the model and the view and coordinates the flow of data between them.

Ch18

- 🌟 Design Mistakes

> - Kitchen sink
> - Overdesigning
> - Non-standard interaction
> - Lack of speed

Ch19

- Software Quality

> - Durability
> - Serviceability

> - Aesthetics
> - Perception

- 3 Costs of Quality

> - Prevention costs
> - Internal Fail
> - External Fail

Ch20

- Error vs. Defect

> - Error : before release, low cost
> - Defect : after release. high cost

- Non-Formal Technical Review

> - Desk check
> - Casual Meeting
> - Pair Programming

- FTR Objectives

> - Walkthrough
> - Inspection

- FTR Guideline

> - Review the product not the producer
> - Set an agenda
> - Limit debate
> - Take written notes
> - Review early reviews

Ch21

- 🌟 Six Sigma:3.4 defects / millon

> - Define
> - Measure
> - Analyze
> - Improve
> - Control

- SQA Goals

> - Requirements quality
> - Design quality
> - code quality

> - Quality control effectiveness

Ch22

- What test shows

> - Errors
> - Requirements Conformance
> - Performance
> - Indication of quality

- Verification vs. Validation

> - Are we building the product right vs. Are we building the right product

- Debugging Techniques

> - Brute force
> - Backtracking
> - Induction
> - Deduction

Ch23

- 🌟 Good Test

> - has a high probability of finding an error
> - not redundant
> - "best of breed"
> - neither too simple nor too complex

- Black box testing

> - Graph-based
> - Equivalence Partition
> - Boundary Value Analysis
> - Comparison Testing
> - Orthogonal Array Testing

Ch24

- OO Testing

> - Unit Testing
> - Integration Testing
>   - Thread-based
>   - Use-based
>   - Cluster

> - Validation Test
> - Fault-based test
> - Random Test
> - Inter Class Test

- Class Model Consistency -> CRC
- Classes

> - Attributes
> - Operations
> - Messages

Ch25

- Testing Quality Dimension

> - Content
> - Funciton
> - Stucture
> - Usability
> - Navigability
> - Performance
> - Compatibility
> - Interoperability
> - Security

Ch26

- Mobile Usability Elements

> - Functionality
> - Information Architecture
> - Screen Design
> - Input Mechanism
> - Mobile Context
> - Interface Usability
> - Trustworthiness

Ch27

- 🌟 Online Safety Threats

> - Social Media
> - Cloud Computing
> - IoT

> - Mobile Apps

- Security Engineering Analysis

> - Security Requirements Elicitation
> - Security Modeling
> - Measures Design
> - Correctness check

- Analyzing Security Requirements

> - Exposure
> - Threat Analysis
> - Controls

Ch28

- 🌟 Two Formal Modeling and Verification Method

> - Cleanroom software engineering
>   - Increment planning
>   - Requirement Gathering
>   - Box Structure Specification
>   - Formal Design
>   - Correctness Verification
>   - Code gen, inspection and verification
>   - Statistical test planning
>   - Certification
> - Formal methods

- Three Box

> - Black box : top level abstract
> - State box: introduce intermediate states
> - Clear box: bottom level, how to implement

- Formal Specification

> - Desired Properties : consistency, completeness, lack of ambiguity
> - Consistency is ensured by mathematically proving
> - Formal Syntax intepreted in only one way

Ch29

- 变更的来源

> - Business Requirements
> - User Requirements

> - Technical Requirements

- SCM Elements

> - Component
> - Process
> - Construction
> - Human

- SCM Process

> - Version Control
> - Change Control
> - SCM Audit
> - Status Accounting

Ch30

- Metrics

> - Requirements Model Metrics
> - Design Model Metrics: HK 度量
> - Web&Mobile App Metrics
> - Code Metrics: Halstead's
> - Test Metrics
> - Maintainance Metrics: SMI

Ch31

- 🌟 Stakeholder

> - Senior Manager
> - Project Manager
> - Practitioner
> - Customer
> - End User

- 4P

> - People
> - Product
> - Process
> - Project

- 🌟 Team Toxicity

> - Frenzied atomsphere
> - Unclear role
> - Not coordinated process
> - High frastration
> - Continous exposure to failure

Ch32

- 几种度量

> - Project Metrics
> - Process Metrics: DRE
> - Size-Oriented:LOC
> - Function-Oriented: FP

- 质量度量

> - Maintainability
> - Integrity
> - Usability
> - Correctness

- Metrics Guideline

> - Don't use metrics to appraise individuals
> - Don't use metrics to threaten individual
> - Avoid single metric
> - Don't consider metrics that indicates a problem as negative

Ch33

- 几种估计方法

> - Scale based：LOC/FP
> - Process-Based
> - Use-Case Based
> - Empirical Models:COCOMO II

Ch34

- Reason for projects being late

> - Unrealistic deadline
> - Ignorant of risks
> - Inmature Process
> - Miscommunication
> - Underestimate of challenges

> - Irresponsible senior managers

Ch35

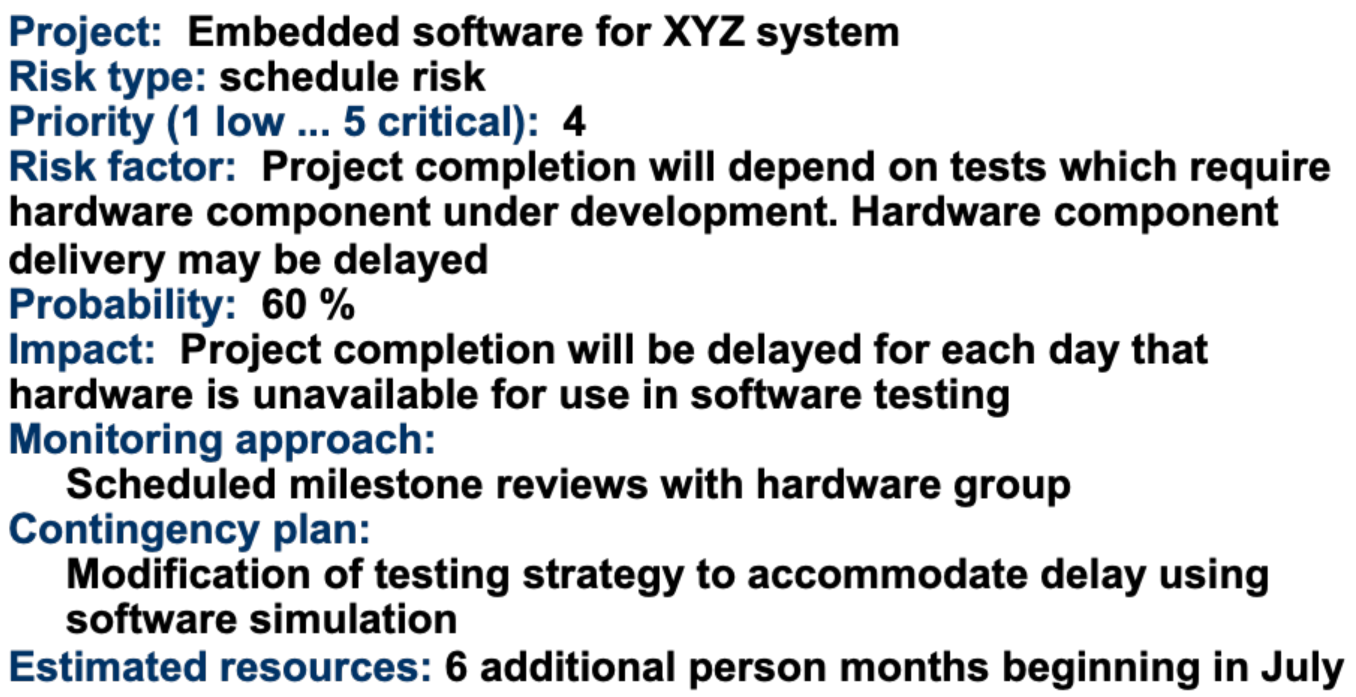
- Risks Components

> - Performance Risk
> - Cost RIsk
> - Support Risk
> - Schedule Risk

- Risk Exposure

> - Risk Identification
> - Risk probability
> - Risk Impact

- ⭐ RMMM

> - Risk Mitigation: How can we avoid the risks
> - Risk Monitering : What factor to track to determine whether the risk is becoming real
> - Risk Management: What contingency plan do we have
>
> 
>
> **Project:** Embedded software for XYZ system
> **Risk type:** schedule risk
> **Priority (1 low ... 5 critical):** 4
> **Risk factor:** Project completion will depend on tests which require hardware component under development. Hardware component delivery may be delayed
> **Probability:** 60 %
> **Impact:** Project completion will be delayed for each day that hardware is unavailable for use in software testing
> **Monitoring approach:**
> Scheduled milestone reviews with hardware group
> **Contingency plan:**
> Modification of testing strategy to accommodate delay using software simulation
> **Estimated resources:** 6 additional person months beginning in July

Ch36

- ⭐ Software Reengineering Steps

> - Inventory Analysis

> - Document Restructuring
> - Reverse Engineering
> - Code Restructuring
> - Data Restructuring
> - Forward Engineering

Test Strategies

PRINCUSICC

- **P**erformance
- **R**egression
- **I**ntegration
- **N**avigation
- **C**ontent
- **U**nit
- **S**ecurity
- **I**nterface
- **C**onfiguration
- **C**ertification