Review

1. **Extreme Programming 极限编程**

Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding, and testing,

XP使用面向对象方法作为推荐的开发范型，它包含了策划、设计、编码和测试4个框架活动的规则和实践。

XP defines four framework activities

1) Planning

* Begins with the creation of a set of stories (also called *user stories*)
* Each story is written by the customer and is placed on an index card
* The customer assigns a value (i.e. a priority) to the story
* Agile team assesses each story and assigns a cost
* Stories are grouped to for a deliverable increment
* A commitment is made on delivery date
* After the first increment “project velocity” is used to help define subsequent delivery dates for other increments

2) Design

* Follows the KIS (keep it simple) principle
* Encourage the use of CRC (class-responsibility-collaborator) cards (Chapter 8)
* For difficult design problems, suggests the creation of “spike solutions”—a design prototype
* Encourages “refactoring”—an iterative refinement of the internal program design
* Design occurs both before and after coding commences

3) Coding

* Recommends the construction of a series of unit tests for each of the stories before coding commences
* Encourages “pair programming”
  + Mechanism for real-time problem solving and real-time quality assurance
  + Keeps the developers focused on the problem at hand
* Needs continuous integration with other portions (stories) of the s/w, which provides a “smoke testing” environment (Chapter 13)

4) Testing

* Unit tests should be implemented using a framework to make testing automated. This encourages a regression testing strategy.
* Integration and validation testing can occur on a daily basis
* Acceptance tests, also called customer tests, are specified by the customer and executed to assess customer visible functionality
* Acceptance tests are derived from user stories

1. **Software project management 项目管理**

Software project management is an umbrella activity within software engineering. It begins before any technical activity is initiated and continues throughout the definition, development, and support of computer software.

Effective software project management focuses on the four P’s: people, product, process, and project

软件项目管理是软件工程的保护性活动。它先于任何技术活动之前开始，且持续贯穿于整个计算机软件的定义、开发和维护之中。

有效的软件项目管理集中于4个P上，即人员、产品、过程和项目，它们的顺序不是任意的。

Effective software project management focuses on four items (in this order)

* + The people
    - Deals with the cultivation of motivated, highly skilled people
    - Consists of the stakeholders, the team leaders, and the software team
  + The product
    - Product objectives and scope should be established before a project can be planned
  + The process
    - The software process provides the framework from which a comprehensive plan for software development can be established
  + The project
    - Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
    - In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

1. **Domain specific architectural models** 
   * Structural models
     + Organized collection of components
   * Framework models
     + Abstract to repeatable architectural patterns
   * Dynamic models
     + Behavioral (dynamic) aspects of structure
   * Process models
     + Business or technical process to be built
   * Functional models
     + Functional hierarchy of the system
2. **Verification 验证**

Verification refers to the set of tasks that ensure that software correctly implements a specific function.

Verification: "Are we building the product right?"

验证是指确保软件正确地实现某一特定功能的一系列活动。

验证：我们在正确地构造产品吗？

1. **Fault, Error , Bug 故障，错误，隐错**

Defect and fault: a quality problem that is discovered after the software has been released to end users

Error: a quality problem that is discovered before the software is released to end users

缺陷和故障是同义词，两者都是指在软件发布给最终用户后发现的质量问题

错误指在软件交付给最终用户之前发现的质量问题

隐错

1. **Objects (in Object Oriented Design)**

对象是特定类的实例并继承类的属性和控制属性的操作。

1. **Reliability**

[ISO 9126 Quality Factors]

The amount of time that the software is available for use as indicated by the following subattributes: maturity, fault tolerance, recoverability.

软件可用的时间长度，由以下子属性表征：成熟性、容错性和易恢复性。

[Garvin]

Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error-free?

软件是否无误的提供了所有的特性和能力，当需要时，它是否是可用的，是否无错地提供了功能？

* A simple measure of reliability is *mean-time-between-failure* (MTBF), where

MTBF = MTTF + MTTR

* The acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to-repair*, respectively.
* *Software availability* is the probability that a program is operating according to requirements at a given point in time and is defined as

Availability = [MTTF/(MTTF + MTTR)] x 100%

1. **Integration testing**

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested components and build a program structure that has been dictated by design.

集成测试是构造软件体系结构的系统化技术，同时也是进行一些旨在发现与接口相关的错误的测试。其目标是利用已通过单元测试的构件建立设计中描述的程序结构。

There are 2 different strategies for integration testing of OO system:

**1) Thread-based testing** integrates the set of classes required to respond to one input or event for the system

**2)** **Use-based testing** begins the construction of the system by testing those classes (called independent classes) that use very few (if any) of server classes. After the independent classes are tested, the next layer of classes, called dependent classes

**3) Cluster testing** is one step in the integration testing of OO software, it defines a cluster of collaborating classes (determined by examining the CRC and object-relationship model) is exercised by designing test cases that attempt to uncover errors in the collaborations.

Cluster testing integrates the set of classes required to demonstrate one collaboration

1. **Regression testing**

regression testing is the re execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects

Regression testing is the activity that helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors

Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated capture/playback tools

1. **Black box testing**

Black-box testing, also called behavioral testing, focuses on the functional requirements of the software. That is, black-box testing techniques enable you to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class for errors than white-box methods.

Black-box testing attempts to find errors in the following categories:

(1) incorrect or missing functions;

(2) interface errors;

(3) errors in data structures or external database access;

(4) behavior or performance errors,

(5) initialization and termination errors.

黑盒测试，也称行为测试，侧重于软件的功能需求。即，黑盒测试使软件工程师能设计出将测试程序所有功能需求的输入条件集。黑盒测试并不是白盒测试的替代品，而是作为发现其他类型错误的辅助方法。

黑盒测试试图发现以下类型的错误：（1）不正确或遗漏的功能；（2）接口错误；（3）数据结构或外部数据库访问错误；（4）行为或性能错误；（5）初始化和终止错误。

1. **White box testing**

White-box testing, sometimes called glass-box testing, is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases.

Using white-box testing methods, you can derive test cases that

(1) guarantee that all independent paths within a module have been exercised at least once,

(2) exercise all logical decisions on their true and false sides,

(3) execute all loops at their boundaries and within their operational bounds,

(4) exercise internal data structures to ensure their validity.

白盒测试，有时也称为玻璃盒测试，是一种测试用例设计方法，它利用作为构件层设计的一部分所描述的控制结构来生成测试用例。利用白盒测试方法导出的测试用例可以：（1）保证一个模块中的所有独立路径至少被执行一次；（2）对所有的逻辑判定均需测试取真和取假两个方面；（3）在上下边界及可操作的范围内执行所有的循环；（4）检验内部数据结构以确保其有效性

1. **Fault tolerance (or tolerant)**
2. **Fault detection**
3. **Recovery block**
4. **Waterfall model**

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment culminating in ongoing support of the completed software

瀑布模型，又被称为经典生命周期，它提出了一个系统的、顺序的软件开发方法，从用户需求规格说明开始，通过计划、建模、构建和部署的过程，最终提供一个完整的软件并提供持续的技术支持

The process of the waterfall:

Communication—planning—modeling—construction—deployment

**When to use the waterfall model:**

* Requirements are very well known
* Product definition is stable
* Technology is understood
* New version of an existing product
* Porting an existing product to a new platform.

Strengths:

* Easy to understand, easy to use
* Provides structure to inexperienced staff
* Milestones are well understood
* Sets requirements stability
* Good for management control (plan, staff, track)
* Works well when quality is more important than cost or schedule

Deficiencies:

* All requirements must be known upfront
* Deliverables created for each phase are considered frozen – inhibits flexibility
* Can give a false impression of progress
* Does not reflect problem-solving nature of software development – iterations of phases
* Integration is one big bang at the end
* Little opportunity for customer to preview the system (until it may be too late)

1. **Software prototyping**

The prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.

Prototyping can be used as a stand-alone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models noted in this chapter.

The prototyping paradigm begins with communication.

The prototype serves as mechanism for identifying software requirements

Although problems can occur, prototyping can be an effective paradigm for software engineering.

Evolutionary models, such as prototyping and the spiral model produce incremental work products quickly.

1. **Safety critical systems**
2. **Systems engineering**

Software engineering occurs as a consequence of a process called system

engineering. Instead of concentrating solely on software, system engineering

focuses on a variety of elements, analyzing, designing, and organizing those

elements into a system that can be a product, a service, or a technology for the

transformation of information or control.

1. **Software quality assurance**

Software assurance is a software engineering umbrella activity that is applied at each step in the software process. SQA encompasses procedures for the effective application of methods and tools, oversight of quality control activities such as technical reviews and software testing, procedures for change management, procedures for assuring compliance to standards, and measurement and reporting mechanisms.

软件质量保证是在软件过程中的每一步都进行的“普适性活动”。SQA包括：对方法和工具有效应用的规程，对诸如技术评审和软件测试等质量控制活动的监督，变更管理规程，保证符合标准的规程，以及测量和报告机制

1. **Software maintenance**

Software maintenance is the most costly phase of the software life cycle. Maintenance of the software conﬁguration nearly always means maintenance of the procedural design representation

1. **Software configuration management**

Software configuration management(SCM), alse called change management, is a set of activities designed to manage change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.

软件配置管理也称为变更管理，是一组管理变更的活动。它通过下面的方式来管理变更：识别可能发生变更的工作产品，建立这些工作产品之间的关系，制定管理这些工作产品的不同版本的机制，控制所施加的变更，审核和报告所发生的变更

1. **Walk-through, FTR (formal technical review)**

A formal technical review is a software quality assurance activity performed by software engineers (and others).

The objectives of the FTR are

(1) to uncover errors in function, logic, or implementation for any representation of the software;

(2) to verify that the software under review meets its requirements;

(3) to ensure that the software has been represented according to predeﬁned standards;

(4) to achieve software that is developed in a uniform manner;

(5) to make projects more manageable. In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation.

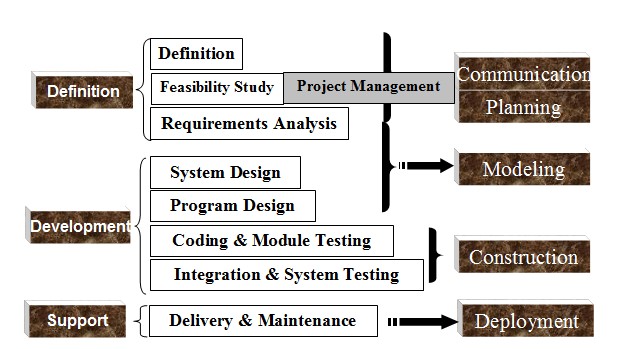
The FTR also serves to promote backup and continuity because a number of people become familiar with parts of the software that they may not have otherwise seen.

1. **Software life cycle**

**定义时期** 基本任务是：确定软件开发工程的总目标；研究该项目的可行性；分析确定客户对软件产品的需求；估算完成该项目所需的资源和成本，并且制定工程进度表。这个时期的工作称为系统分析，由系统分析员负责完成。

**开发时期** 基本任务是：具体设计和实现在前一个时期定义的软件，软件开发时期它通常由下述4个阶段组成：总体设计(又称为概要设计、结构设计)；详细设计；编码和单元测试；综合测试。其中前两个阶段又称为系统设计，后两个阶段又称为系统实现。由系统设计员和编程人员、测试人员负责完成。

**软件维护时期** 基本任务是：对交付使用的系统进行升级和后期维护管理，由系统维护员负责完成。



1. **Validation**

Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Validation:” Are we building the right prosuct?”

确认指的是确保开发的软件可追溯到客户需求的一系列活动

确认：我们在构造正确的产品吗？

1. **Partition testing**

Reduces the number of test cases required to test a class in much the same way as equivalence partitioning for conventional software. Input and output are categorized and test cases are designed to exercise each category.

划分测试减小测试特定类所需的测试用例数量。对输入和输出进行分类，设计测试用例以检查每个分类。

1.  state-based partitioning

categorize and test operations based on their ability to change the state of a class

1.  attribute-based partitioning

categorize and test operations based on the attributes that they use

1.  category-based partitioning

categorize and test operations based on the generic function each performs

1. **Fault avoidance**
2. **Requirements analysis**

Requirement analysis results in the specification of software’s operational characteristics, indicates software’s interface with other system elements, and establishes constraints that software must meet.

需求分析产生软件工作特征的规格说明，指明软件和其他系统元素的接口，规定软件必须满足的约束。

The requirements modeling action results in one or more of the following types of models:

* + - * Scenario-based models of requirements from the point of view of various system “actors”.
      * Data models that depict the information domain for the problem.
      * Class-oriented models that represent object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements.
      * Flow-oriented models that represent the functional elements of the system and how they transform data as it move through the system.
      * Behavioral models that depict how the software behaves as a consequence of external “events”.

1. **Software engineering**

* The establishment and use of sound engineering principles in order to obtain economically feasible software that works efficiently on real machines [F.Bauer, 1969]

建立和使用一套合理的工程原则，以便经济地获得可靠的、可以在实际机器上高效运行的软件

* More than a discipline or a body of knowledge: It is a way of approaching a problem [S. Whitmire]
* The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; I.e. the application of engineering to software.  
  (2) The study of approaches as in (1) above. [IEEE 1993]

1. **GUI design principles**