第一章：概述

1. 软件是**程序（指令）、数据（结构）和文档**的集合。

2. 软件工程意味着对软件的**开发、运行、维护**等采用**系统的、规范、可度量的**方法。也就是说，将工程应用到软件中。

3. 任何解决问题的技术必须有两部分**，分析问题以确定其性质，然后合成解决方案**。

4. 客户是支付软件系统开发费用的公司、组织或个人。客户端是建立软件系统的公司、组织或个人。用户是实际使用系统的人或群体。

5. 对于任何需要工程化的实体，我们必须进行**分析、设计、构建、验证、管理**。

第二章：

1. 我们可以把一系列有序的任务想象成一个**过程**，一个包含**活动、约束和资源**的步骤。

2. 当过程涉及到某种产品的建造时，我们有时将过程称为生命周期。

3. 软件产品的生命周期包括**概念、实施、交付、使用、维护**。

4. 瀑布模型包括**需求分析、系统设计、程序设计、编码、单元与集成测试、系统测试、验收测试，使用与维护**等步骤。

5. 验证确保系统已经实现了所有的需求，但是验证确保每个功能都正常工作。

6. V模型更**明确地显示**了在瀑布模型中**隐藏的迭代和重新工作。**

7. 螺旋模型将**开发活动与风险管理**结合起来，以最小化并**控制风险**。

8. **原型**是一个部分开发的产品，使客户和开发者能够检查预设系统的某些方面，并决定它是否适合最终产品。

第三章：

1. **项目进度表**通过列举项目的各个阶段并将每个**阶段**分解为具体的任务或活动，描述了特定项目的软件开发周期。

2. 进度表是一个**时间线**，显示活动的开始和结束时间，以及相关开发产品的准备情况。

3. **交付物**是客户在项目开发过程中期望看到的**项目Items** 。

4. **活动**是在一段时间内发生的项目的一部分，而**里程碑**是活动的完成-一个特定的时间点。

5. **甘特图**描绘了项目作为一组离散工作的状态。discrete pieces of work.

6. 实时或实际时间 **Real time or actual time** 是完成活动所需的预计时间。可用时间是日程表中活动完成的可用时间。**slack time or float time**是可用时间和实际时间的差值。

7. **CPM**是一条在每个节点的空闲时间为零的路径。

8. 甘特图用于描绘项目的进度。 **schedule**

第四章：

1. 需求是软件行为的表达。**Expression**

2. 捕获需求的过程有四个步骤，分别是需求收集、分析、规格定义和验证。 **Elicitation, analysis, specification, validation**

3. 需求从七个系统利益相关者中获取，他们是**客户、用户、领域专家、市场研究员、 domain experts, market researcher律师或审计员auditors、软件工程师。**

4. 可能的需求来源包括利益相关者的需求和需求**wants and needs,** 、**领域模型、当前情况模型、可复用的需求、建议的需求类型、现有的文档、当前的组织和系统。**

5. 有两种类型的需求文档，它们是**需求定义**和**需求规格。**

6. 在数据流图(DFD)中，有四个核心构造，它们是**过程、数据流、数据存储、行为者。** process, data flow, data store, actor

7. 在用例图(UCD)中，一个大的框代表**系统边界**。

8. 原型的两种方法是抛弃原型和进化原型。**throwaway prototype, evolutionary prototype**

第五章：

1. 设计是确定如何实现客户所有需求的创新过程；**结果**计划称为设计。

2. 软件架构由三部分组成，它们是**组件、连接器、约束**。**components, connectors, constrains**

3. 在管道和过滤器样式中，**过滤器的功能是通过一系列的数据转换组件传递输入数据，管道仅仅是从一个过滤器传输数据到下一个过滤器，而不改变数据。**

4. 在客户-服务器架构中，**服务器组件提供服务，客户端通过请求/回复协议访问这些服务**。

5. 在对等架构中，每个对等体都**作为自己的进程执行**，并作为其他对等组件的**客户端和服务器。**

6. 在发布-订阅架构中，订阅组件对事件表示兴趣，当另一个组件发布该事件已经发生时，订阅组件被通知。

7. 存储库样式由两种类型的组件组成，一种称为**中央数据存储，另一种称为数据访问组件**。

第六章：

1. 设计模块意味着决定如何在模块级别设计**单个组件**，以便开发人员可以编写**实现设计**的代码。

2. 本章可以帮助你理解为什么某些设**计原则和约定**适用，并帮助你决定何时应用它们。

3. 在实践中，**架构设计阶段的结束和模块设计阶段**的开始之间没有**明确的界限**。 **sharp boundary**

4. 设计原则是将我们**系统所需的功能和行为分解为模块**的指导原则。

5. 模块性，**也称为关注点分离，是将系统的各个无关方面保持分离的原则。**

6. 模块是**时间性temporal**的，因为模块的数据和函数之间的关系只是因为它们**在执行时被同时使用**。

第七章：

1. 即使在编写代码本身时，也通常需要许多人参与，需要大量的协作和协调。**coorperation and coordination**

2. 许多公司坚持他们的代码遵守**样式、格式和内容标准**，**style, format** and **content** standards, 这样代码和相关标准对于阅读他们的每个人都是清晰的。

3. 无论使用何种语言，每个**程序**组件都至少涉及到三个主要方面，即**控制结构、算法和数据结构。**

4. **程序文档**被认为是一组书面描述，向读者解释程序的工作方式和执行方式。**内部文档**是**在代码中编写的描述性材料**，所有其他文档都是外部文档。

第八章：

1. 当人在执行某些软件行为时出错时，就会发生故障fault ，这种错误被称为错误，因此，故障是系统的内在视图，通过开发人员的眼睛看到的。

2. 失败是系统所需行为的**Departure** **偏离**。**developer** 因此，失败是一个外部视图：用户看到的问题。

3. 故障识别是**确定**故障或故障导致失败的过程，故障修正或移除是对系统进行更改以移除故障的过程。

4. 当组件的**算法或逻辑algorithm or logic**对给定输入没有产生正确的输出时，就会发生算法故障。

5. 计算和精度故障Computation and precision 发生在公式的实现错误或者没有计算到所需的小数位数。**decimal**

places.

6. 当文档与程序实际执行的内容不匹配时，我们说程序有**文档错误。**

7. 压力或过载故障发生在数据结构被填充超过其指定的**容量**。

8. 功能测试评估系统以确定需求规格描述的函数是否由集成系统实际执行。

9. 性能测试将系统**与剩余的软**件和硬件要求进行比较。

10. 验收测试根据客户的需求描述进行检查，这个测试与客户一起完成。

11. 安装测试运行以确保系统仍然如期运行。

12. 我们从外部观察测试对象，作为一个封闭的或黑箱，其内容未知，我们的测试向封闭的箱子**提供输入**，并注意到产生了**什么输出**。

13. 我们将测试对象视

为一个打开的或白盒，我们看到内部，知道在测试对象内部会发生什么。

第十四条：

1. 测试点或测试案例是程序中使用的特定**输入数据的**选择。

2. 在自底向上的集成测试中，我们应开发一个**组件驱动器**，将测试案例传递给要测试的组件。

3. 在自顶向下的集成测试中，我们编写一个组件来模拟缺失组件的活动。类似Stub

第九章：

1. 系统配置是交付给特定客户的**系统组件的集合**。

2. 软件的可靠性是系统在给定条件下在给定时间**内无故障运行**的可能性。

3. 软件的可用性是系统在给定时间点根据规**格成功运行**的概率。

4. 软件的可维护性是，对于给定的使用条件，可以在规定的时间间隔内，并使用规定的程序和资源进行维护的概率。

5. 在开发环境中进行的**测试被称为alpha测试，而在客户环境中进行的测试被称为beta测试。**

第十章：

用户运行主要的系统功能，操作员执行支持主要工作的系统功能。

第十一章：

1. 运营中的系统进行的任何改变都被认为是维护。

2. S-system由规范正式定义并可从规范中推导出来。

3. P-system非常抽象，实现几乎是不切实际和不可能的。

4. 为了控制日常的系统功能，我们在维护团队中解决故障问题。这种类型的维护被称为修正维护。

5. 假设现有的数据库管理系统升级到了新版本，这种维护被称为适应性维护。

6. 如果客户想要添加新的功能，这种类型的维护被称为完善性维护。

7. 典型的开发项目需要一到两年的时间，但是需要额外的五到六年的维护时间。

8. 一个项目的一半以上的编程资源都用在了维护上。

9. 根据80-20规则：20%的工作量用在了开发上，80%的工作量用在了维护上。