```yaml
title: 大学课程 | 软件工程知识点
tags:
  - 软件工程
  - 大学课程
categories:
  - 学习笔记
abbrlink: 38368
reward: true
copyright: true
date: 2019-07-10 20:39:36
cover: https://npm.elemecdn.com/justlovesmile-img/1584111722-u198426915126376500fm26gp0.jpg
top_img: https://npm.elemecdn.com/justlovesmile-img/1584111722-u198426915126376500fm26gp0.jpg
```

作者博客：Justlovesmile's BLOG

大二软件工程课程笔记

软件工程要点

1. 软件是指令的集合，数据结构和软件描述信息的集合
2. 软件与硬件的区别：软件不会磨损，但会退化，退化的根本原因是不断变更
3. 软件工程是（1）将系统化的，规范的，可量化的方法应用于软件的开发，运行和维护.（2）对（1）中方法的研究
4. 工程化：系统化的，规范的，可量化的
5. 软件工程是一种层次化的技术，包含工具，方法，过程，质量关注点。
6. 软件工程的三要素：工具，方法，过程
7. 软件过程是工作产品构建是所执行的一系列活动，动作和任务的集合
8. 五种通用过程框架活动：沟通，策划，建模，构建，部署
9. 软件工程整体实践的原则：存在价值；保持简洁；保持愿景；关注使用者；面向未来；提前计划复用；认真思考
10. 四种过程流：线性过程流；迭代过程流；演化过程流；并行过程流；
11. 惯用过程模型：瀑布模型；V模型；增量过程模型；原型模型，螺旋模型；并发模型
12. 专用过程模型：基于构件；形式化方法模型；面向方面
13. 统一建模语言UML
14. UP统一过程的五个阶段：起始阶段；细化阶段；构建阶段；转换阶段；生产阶段
15. UP的五个阶段不是顺序进行，而是阶段性的并发进行
16. 敏捷原则（12条）
17. 普遍存在的变更是敏捷的基本动力
18. 需求工程是一个软件工程的动作，开始于沟通并持续到建模活动
19. 需求工程的7项任务：起始，获取，细化，协商，规格说明，确认，管理
20. 分析模型的作用：为基于计算机的系统提供必要的信息，功能和行为域的说明
21. 分析模型的元素：基于场景的元素；基于类的元素；行为元素数据流元素；
22. 需求建模动作结果：场景模型；面向类的模型；基于行为和模式的模型；数据模型；面向流的模型
23. 域分析：识别，分析和详细说明某个特定应用领域的共同需求以便确定可以在整个邻域内复用的对象
24. 需求建模的方法：结构化分析；面向对象分析（UML和UP）
25. 用例间的关系：包含关系（一个用例总是使用另一个用例的功能）；扩展关系；泛化关系（子类与夫类的关系）
26. UML活动图：两端为半圆的矩形—特定的系统功能；箭头—通过系统的流；菱形—分支；实水平线—并行发生的活动
27. 泳道图：参与者职责由纵向分割图中的并行条表示

28. 类的分类：实体类；边界类；控制类

29. 类间关系：关联；继承；依赖

30. 面向对象的目的是封装，但仍保持对的数据以及对数据的操作

31. 潜在类的特征：保留信息；所需服务；多个属性；公共属性；公共操作；必要需求

32. CRC模型：是表示类的标准索引卡片的集合

33. CRC评审模型

34. UML状态图：箭头—状态转移

35. 时序图（顺序图）

36. 四种设计模型：构件级设计；接口设计；体系结构设计；数据/类设计；

37. 设计概念包括：抽象，体系结构，模式，关注点分离，模块化，信息隐蔽，功能独立。求精，方面，重构，面向对象，设计类（完整性与充分性，原始性，高内聚性，低耦合性），依赖倒置，测试设计

38. 软件体系结构：程序或计算系统的软件结构是指系统的一个或多个结构，它包括软件构建，构建的外部可见属性以及他们之间的相互关系

39. 设计是体系结构的一个实例

40. 体系结构的风格包括：（1）完成系统需要的某种功能的一组构件；（2）能使构件间实现“通信，合作和协调”的一组连接件（3）定义构件如何集成为系统的约束（4）语义模型，能使设计者通过分析系统组成成分的已知属性来理解西戎的整体性质

41. 体系结构风格的分类：以数据为中心的体系结构；数据流体系结构；调用和返回体系结构；面向对象体系结构；层次体系结构

42. 构件：系统中模块化的，可部署的和可替换的部件，该部件封装了实现对外提供一组接口

43. 基本设计原则：开闭原则；Liskov替换原则；依赖倒置原则；接口分离原则；发布复用等价性原则；共同封装原则，共同复用原则

44. 内聚性：功能内聚；分层内聚；通信内聚

45. 耦合性：内容耦合；公共耦合；外部耦合；控制耦合（由强到弱）

46. 构件3C模型：概念，内容，环境

47. 用户界面黄金原则：（1）把控制权交给用户（2）减轻用户的记忆负担（3）保持界面一致

48. 软件测试策略：单元测试 集成测试 确认测试 系统测试

49. 集成测试包括：自顶向下集成；自底向上集成；回归测试；冒烟测试

50. 面向对象的集成测试：基于线程的测试；基于使用的测试；簇测试

51. α测试和β测试的区别：α测试是由有代表性的最终用户在开发者的场所进行。即α测试是在受控环境下进行。而β测试在一个或多个最终用户场所进行。与α测试不同，开发者通常不在场

52. 系统测试包括：恢复测试；安全测试；压力测试；性能测试；部署测试

53. 调试方法：蛮干法；回溯法；原因排除法

54. 白盒测试：（结构化测试）
（1）逻辑覆盖
（2）路径覆盖：①流图：箭头—边/连接/控制流；边和结点限制的区域—域；包含条件的结点—判定结点②环复杂性：（1）V(G)=边数-节点数+2（2）V(G)=判定结点数+1（3）V(G)=域数③生成基本测试用例：（1）以设计或源代为基础画出相应的流图（2）确定所得流图的环复杂性（3）确定线性独立路径的基本集合（4）准备测试用例，强制执行基本集合中的每一条路径

55. 黑盒测试：（行为测试/功能测试）白盒测试在测试早期执行，黑盒测试倾向于在测试后期
（1）等价类法
（2）边界值法
（3）错误猜测法

56. 软件配置：在软件过程中产生的所有信息项

57. 软件配置管理：一组用于在计算机软件的整个生命周期内管理变更的活动

58. 软件过程输出信息可以分为：计算机程序；文档；数据或内容

59. 系统工程第一定律：不管你处在系统生命周期的什么阶段，系统都可能发生变更，并且在整个生命周期中将会持续不断的提出变更的要求

60. 配置管理系统的元素：构件元素；过程元素；构建元素；人员元素；

61. 基线：已经通过正式评审和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制规程才能修改它

62. SCM中心存储库：是一组机制和数据结构，它使软件团队可以有效地管理变更

63. SCM特征：版本控制；依赖性跟踪和变更管理；需求跟踪；配置管理；审核跟踪

64. 软件项目管理的4P：人员，产品，过程，项目

65. W5HH原则

66. 软件度量：过程度量，项目度量，产品度量

67. 软件测试：
    （1） 面向规模度量：LOC（代码行）
    （2） 面向功能度量：FP（功能点）
    （3） 面向对象度量
    （4） 面向用例度量

68. 软件质量：测试指标：正确性；可维护性；完整性；可用性；
    （1） 正确性：每千行代码的缺陷数
    （2） 可维护性：平均变更时间MTTC
    （3） 完整性：=∑(1-(危险性×(1-安全性)))

69. 缺陷排除效率：
    （1） DRE=E/(E+D)
    （2） E是软件交付给最终用户之前发现的错误数
    （3） D是软件交付之后发现的错误数

70. 三类主要软件工程资源：人员，可复用的软件构件，开发环境（硬件和软件工具）
    （1） 对每类资源都要说明四个特征：资源描述，可用性说明，何时需要资源，使用资源的持续时间

71. 基于问题估算：
    （1） S=（Sopt+4Sm+Spess）/6
    （2） Sopt乐观值，Sm可能值，Spess悲观值

72. 基于LOC的估算：
    （3） LOC/pm--人月

73. 基于FP的估算：
    （4） FPestimated=总计×（0.65+0.01×∑Fi）

软件工程简答题及答案

1.简述软件的定义及特征

软件是：计算机系统中与硬件相互依存的另一部分，它包括程序，数据及其相关文档的完整集合。
（1）指令的集合（计算机程序），通过执行这些指令可以满足预期的特性、功能和性能需求；
（2）数据结构，使得程序可以合理利用信息；
（3）软件描述信息，用来描述程序的操作和使用。

- 软件的特征是：
  - 软件不会“磨损”，但会退化，退化的根本原因是不断变更；
  - 软件是开发、设计出来的，不是生产出来的；
  - 大多数软件是按照实际客户要求定制的。

2.简述软件工程的定义及软件过程的5种框架活动

- 软件工程是：
  （1）将系统化的、规范的、可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件；
  （2）对（1）中所述方法的研究。
- 软件工程过程框架通常包含以下5个活动：
  - 沟通、策划、建模、构建、部署。

3.画图说明软件过程流的各种类型

略

4.画图说明软件过程的增量模型及适用情形和特点

略

- 适用情形：初始的软件需求有明确的定义，但是整个开发过程却不宜单纯运用线性模型。迫切需要为用户迅速提供一套功能有限的软件产品，然后在后续版本中再进行细化和扩展功能。
- 特点：增量模型综合了线性过程流和并行过程流的特征。每个线性序列生产出软件的可交付增量且第一个增量往往是核心产品，满足了基本的需求。客户使用后进行评估，根据评估结果制定下一个增量计划。每一个增量的交付都会重复这一过程，直到最终产品产生。

5.画图说明软件过程的原型模型及适用情形和特点

略

- 适用情形：客户定义了软件的一些基本任务，但是没有详细定义功能和特性需求。开发人员可能对算法的效率、操作系统的适用性和人机交互的形式等情况并没有把握。

- 特点：在大多数项目中，构建的第一个系统很少是好用的，可能太慢了、太大了、太难用了，或者同时具备上述三点。一般作为被丢弃的系统。开发者没有考虑整体软件质量和长期的可维护性。软件工程师经常会使用不合适的操作系统或程序设计语言或低效的算法。但是原型开发对于软件工程来说仍是有效的范型。

6.画图说明统一过程的各个阶段

略
UP：

- 起始阶段，细化阶段，构建阶段，转换阶段，生产阶段
- 五个UP阶段不是顺序进行，而是阶段性地并发进行。（可能在构建转换生产同时，下一个软件增量的工作已经开始）

7.简述敏捷原则

（1）我们最优先要做的是通过尽早、持续交付有价值的软件来使客户满意。
（2）即使在开发的后期，也欢迎需求变更。敏捷过程利用变更为客户创造竞争优势。
（3）经常交付可运行软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。
（4）在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。
（5）围绕有积极性的个人构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。
（6）在团队内部，最富有效果和效率的信息传递方法是面对面交谈。
（7）可运行软件是进度的首要度量标准。
（8）敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够长期保持稳定的开发速度。
（9）不断地关注优秀的技能和好的设计会增强敏捷能力。
（10）简单——使不必做的工作最大化的艺术——是必要的。
（11）最好的架构、需求和设计出自于自组织团队。
（12）每隔一定时间，团队会反省如何才能更有效地工作，并相应调整自己的行为。

8.简述需求建模的模型

（1）场景模型：出自各种系统“参与者”观点的需求。
（2）面向类的模型：表示面向对象类（属性和操作）的模型，其方式为通过类的协作获得系统需求。
（3）基于行为和模式的模型：描述如何将软件行为看作外部“事件”后续的模型。
（4）数据模型：描述问题信息域的模型。
（5）面向流的模型：表示系统的功能元素并且描述当功能元素在系统中运行时怎样进行数据变换。

9.简述CRC模型的评审步骤

①所有参加评审的人员拿到一部分CRC模型索引卡。拆分协作卡片。
②分类管理所有的用例场景。
③评审组长细致地阅读用例。当评审组长看到一个已命名的对象时，给拥有相应类索引卡的人员一个令牌。
④当令牌传递时，该类卡的拥有者需要描述卡上记录的职责。评审组确定职责是否满足用例需求。
⑤如果记录在索引卡上的职责和协作不能满足用例，就需要修改卡片，包括定义新类。

10.简述行为建模的步骤

（1）评估所有的用例，以保证完全理解系统内的交互顺序；
（2）识别驱动交互顺序的事件，并理解这些事件如何与特定的对象相互关联；
（3）为每个用例生成序列；
（4）创建系统状态图；
（5）评审行为模型以验证准确性和一致性。

11.画图说明从需求模型到设计模型的转换

略

12.简述模块的功能独立及评估标准

通过开发具有“专一”功能和“避免”与其他模块过多交互的模块，可以实现功能独立。
独立性可以通过两条定性的标准进行评估：内聚性和耦合性。内聚性显示了某个模块相关功能的强度；耦合性显示了模块间的相互依赖性。

13.简述重构的定义及重构时的检查要点

重构是使用这样一种方式改变软件系统的过程：不改变代码的外部行为而是改进其内部结构。
在重构软件时，检查现有设计的冗余性、没有使用的设计元素、低效的或不必要的算法、拙劣的或不恰当的数据结构以及其他设计不足。

14.简述体系结构风格描述的4个要素及其分类

四个要素：
（1）完成系统需要的某种功能的一组构件；
（2）能使构件间实现“通信、合作和协调”的一组连接件；
（3）定义构件如何集成为系统的约束；
（4）语义模型，能使设计者通过分析系统组成成分的已知属性来理解系统的整体性质。
分类：以数据为中心的体系结构、数据流体系结构、调用和返回体系结构、面向对象体系结构、层次体系结构。

15.简述构件级设计的7个基本原则

①开闭原则。模块（构件）应该对外延具有开放性，对修改具有封闭性。
②Liskov替换原则。子类可以替换它们的基类。
③依赖倒置原则。依赖于抽象，而非具体实现。
④接口分离原则。多个客户专用接口比一个通用接口要好。

⑤发布复用等价性原则。复用的粒度就是发布的粒度。
⑥共同封装原则。一同变更的类应该合在一起。
⑦共同复用原则。不能一起复用的类不能被分到一组。

16.简述黄金规则中把控制权交给用户的规则

①以不强迫用户进入不必要的或不希望的动作的方式来定义交互模式。
②提供灵活的交互。
③允许用户交互被中断和撤销。
④当技能水平高时可以使交互流线化并允许定制交互。
⑤使用户与内部技术细节隔离开来。
⑥设计应允许用户与出现在屏幕上的对象直接交互。

17.简述黄金规则中减轻用户记忆负担的原则

①减少对短期记忆的要求。
②建立有意义的默认设置。
③定义直观的快捷方式。
④界面的视觉布局应该基于真实世界的象征。
⑤以一种渐进的方式揭示信息。

18.简述黄金规则中保持界面一致的原则

①允许用户将当前任务放入有意义的环境中。
②在完整的产品线内保持一致性。
③如果过去的交互模型已经建立起了用户期望，除非有不得已的理由，否则不要改变它。

19.简述面向对象软件测试中集成测试的3种策略

①基于线程的测试，对响应系统的一个输入或事件所需的一组类进行集成。
②基于使用的测试，通过测试很少使用服务类的那些类开始系统的构建。
③簇测试，借助试图发现协作错误的测试用例来测试协作的类簇。

20.简述压力测试并举例说明

压力测试的目的是使软件面对非正常的情形。压力测试要求以一种非正常的数量、频率或容量的方式执行系统。
例如：（1）在平均每秒出现1~2次中断的情形下，可以设计每秒产生10次中转的测试用例；（2）将输入数据的量提高一个数量级以确定输入功能将如何反应；（3）执行需要最大内存或其他资源的测试用例；（4）设计可能在实际的运行系统中产生惨败的测试用例；（5）创建可能会过多查找磁盘驻留数据的测试用例。

21.简述单元测试中桩模块和驱动模块的作用?

驱动程序只是一个“主程序”，它接受测试用例数据，将这些数据传递给构件，并打印相关结果。
桩程序的作用是替换那些从属于被测构件的模块。桩程序或“伪程序”使用从属模块的接口，可能做少量的数据操作，提供入口的验证，并将控制返回到被测模块。

22.简述测试中症状与原因之间的关系

症状与原因出现的地方可能相隔很远。也就是说，症状可能在程序的一个地方出现，而原因实际上可能在很远的另一个地方。高度耦合的构建加剧了这种情况的发生。

23.简述软件的基线及SCI和项目数据库之间的关系

基线的定义是：已经通过正式评审和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制规程才能修改它。
基线是软件开发中的里程碑，其标志是在正式技术评审中已经获得批准的一个或多个软件配置项的交付。
软件配置项是在软件工程过程中创建的信息。在现实中，是将SCI组织成配置对象，这些配置对象具有自己的名字，并且按类别存储在项目数据库中。配置对象具有一个名称和多个属性，并通过关系来表示与其他配置对象的“关联”。

24.简述选择软件团队结构时应考虑的7个因素

①待解决问题的难度；
②开发程序的规模，以代码行或者功能点来度量；
③团队成员需要共同工作的时间 (团队生存期)；
④能够对问题做模块化划分的程度；
⑤待开发系统的质量要求和可靠性要求；
⑥交付日期的严格程度；
⑦项目所需要的友好交流的程度。

25.简述软件团队的组织范型

①封闭式范型。按照传统的权利层次来组织团队。
②随机式范型。松散地组织团队，团队工作依赖于团队成员个人的主动性。
③开放式范型。试图以一种既具有封闭式范型的控制性，又包含随机式范型的创新性的方式来组织团队。
④同步式范型。依赖于问题的自然划分，组织团队成员各自解决问题的一部分，他们之间没有什么主动的交流。

26.简述如何避免“团队毒性”

为了避免狂乱的工作环境，项目经理应该确保团队可以获取完成工作所需的所有信息；而且，主要目标一旦确定下来，除非绝对必要，否则不应该修改。给予软件团队尽可能更多的决策权，这样能使团队避免挫败。通过理解将要开发的产品和完成工作的人员，以及允许团队选择过程模型，可以避免选择不适当的软件过程，团队本身应该建立

自己的责任机制。团队本身应该建立自己的责任机制，并规定一系列当团队成员未能完成任务时的纠正方法。最后，避免失败的关键是建立基于团队的信息反馈方法和解决问题的技术。