

软工中厅讲座

2018 – 12 – 29

yL给大家讲讲题目

注：

- Ln代表着第几章课件。
- Pm代表课件的第几页。
- 例如：L14P9-15代表着第14章课件Process的第9页到第15页。

Outline

- 软件工程是什么 (Intro)
- 软件项目管理 (Project Management)
- 软件生命周期(Process)
 - 需求(Requirement)
 - 设计(Design)
 - 代码实现(Coding)
 - 测试(Test)
 - 品控(Quality Control)、运维(Operation)

Intro

软工是什么

Intro – 软工的概念

Q1

(1) 20 世纪 60 年代，针对软件危机的问题，提出了软件工程的思想，力图将其它工程领域中行之有效的工程学知识运用到软件开发工作中。下图中：

- 左图是 1979 年美国政府会计总署发布的统计数据，列出了各类软件项目所投入的经费量以及所占的经费比例，其中：支付后从未交付的软件占 28.8%，交付但无法正常使用的软件占 47.3%，交付后需要重大修改的占 19.2%，交付后需少量修改的占 3%，成功投产的占 1.77%。
- 右图取自 2003 年日本的 Nikkei 咨询公司的调研报告，该报告分析了日本 1746 家公司的软件开发项目。本图从软件质量、费用和交付率三个方面统计软件项目成功率，其中：46.4% 的项目满足质量要求，76.2% 的项目按预算执行，54.9% 的项目按期交付。

请结合这两个图表，分析软件危机的问题和软件工程的主要目标。（10 分）

Intro – 软工的概念

- A1 • …【前面的看图回答】。针对这些问题，软件工程的主要目标是建立并使用一套合理的工程原则，从而能够经济地、可靠地、有效地来开发出一套能够投入实地使用的软件。
- 【L1P3】这套原则应该是系统的、经济的和高质量的。

Intro –软工的概念

- Q2
- 一旦我写完程序并让它运行，工作就完成了。
 - 60-80%的精力消耗在软件交给用户之后。
 - 得到程序运行才能评估其质量。
 - 最有效的软件质量保证机制即正式技术评审应该从工程早期就开始应用。
 - 一个成功项目要交付的产品仅仅只有能有效工作的程序。
 - 程序只是软件配置的一部分， 它还包括了许多元素， 例如文档。
 - 软件工程使我们创建大量不必要的文档， 会延缓我们的开发速度。
 - 软件工程并不是创建文件， 而是保证质量。高质量才能减少返工， 从而能更快交货。

Intro –软工的概念

Q3

- 一个大致目标描述足以开始写程序，之后再补充详细信息
 - 一个全面稳定的需求描述往往不可能，但模棱两可的需求描述是灾难。明确需求需要客户和开发人员有效和持续的沟通。
- 项目需求不断变化，软件是活的，所以可以很容易适应变化。
 - 需求变化会引起设计框架的变化，需要额外的资源和设计修改，且变化对成本的影响随时间增加。
- 我们有一本关于构建软件的标准和程序的书，它能为用户提供一切他们要知道的。
 - 这本书很可能存在，它是现在的吗？软件实践人员意识到它的存在吗？它反映了现代软件工程实践吗？它完整吗？它适用吗？它流线型地改善交货时间同时保持对质量的关注吗？在许多情况下？所有这些问题的答案是否定的。

Intro – 软工的概念

- Q4
- 如果落后于计划，可以添加更多的程序员并赶上。
 - 软件开发不是一个机械过程，加人需要花时间教他们工作从而导致开发时间减少，尤其不能在软件开发后期加人。
 - 如果把软件项目外包给第三方，我可以放松，让公司构建它。
 - 一个组织往往不了解软件项目内部的管理和控制。
 - 运行正确的程序就是一个高质量的软件。
 - 正确运行只是高质量软件的一部分。高质量的软件还需要易于维护、易于复用、能够针对需求的变化容易的进行调整、有较好的可拓展性。

Project Management

软工项目管理

Project Management – 软工项目管理 · L15

- 如何管理一个软工项目？
- 人(People)
 - 不同的角色：经理、客户、用户
 - 沟通和交流
- 计划(Schedule)
 - 定义任务、关键路径、里程碑
 - 时间安排图
 - 关键路径计算
 - 4-2-4时间分配：建议整个软件过程的努力分布是40%(分析和设计), 20%(编码), 40%(测试)
- 任务(Task)
 - 足够的区分度、确定的相关性

Project Management – 软工项目管理 · L15

Q5

a) 图 1 给出了一个系统的版本树样例，请分析其版本演化过程。(5 分)



A5

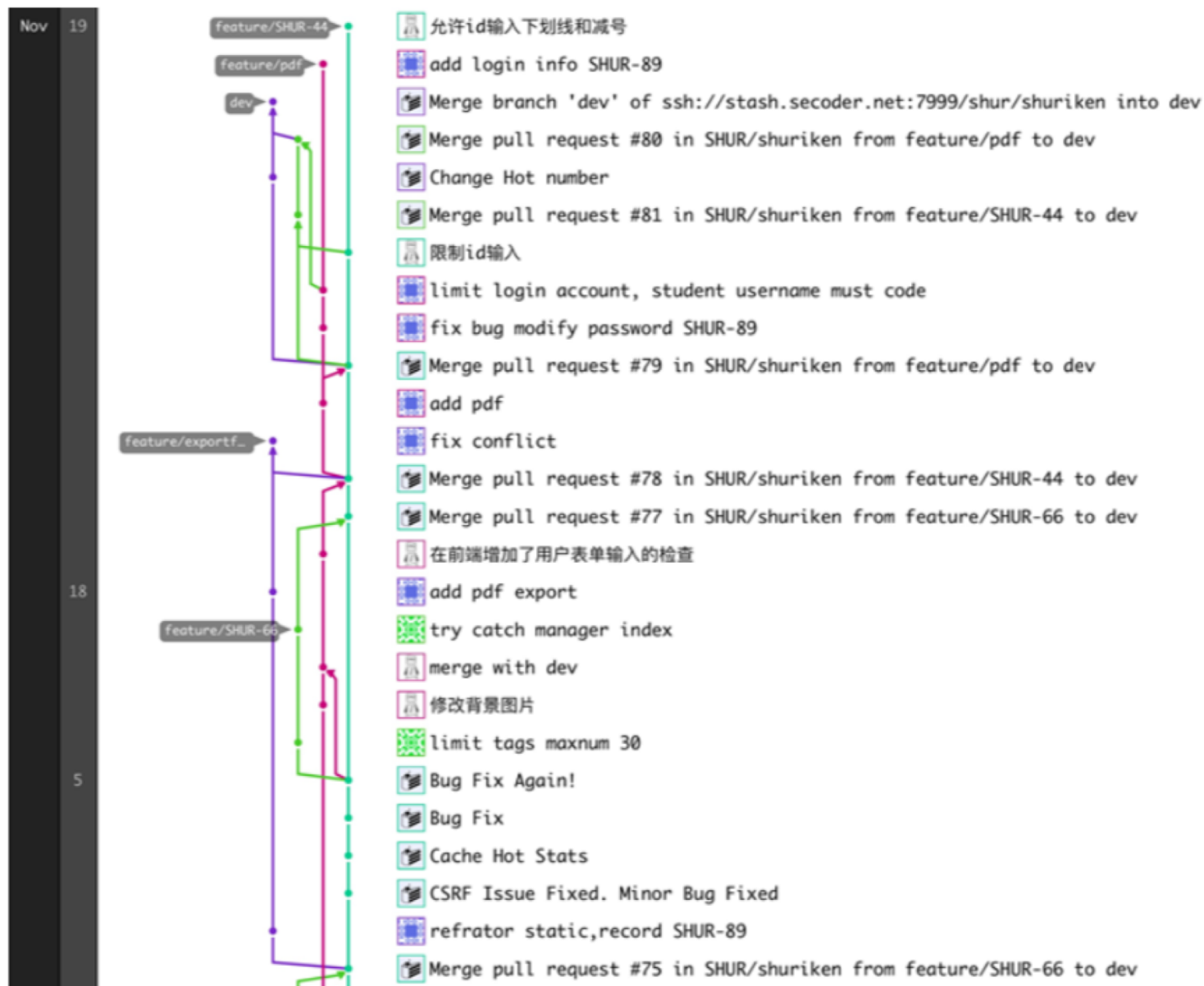
• 参考L2回答。

Project

Q6

b) 图 2 是本学期某大作业版本树片段。请结合版本管理的基本原则，分析其优点和不足之处。(5 分)

L15



Project Management – 软工项目管理 · L15

A6

(b) 优点：

1. 使用了多个分支，便于多个版本或不同模块间的共同开发。
2. commit 次数较多，较好的体现了项目的开发进度
3. 使用了 pull request，更符合多人开发大型项目的做法

缺点：

1. commit 注释中英文混叠，没有一个标准
2. 有用的 commit 注释较短，另有 Bug Fix Again!等无用注释
3. 分支相对比较混乱（并不确定）
4. 11月5号-11月18号之间 commit 次数只有 5-7 次，之后 1 天 commit 十几次，不符合开发规范。（不确定是否属于版本管理）

Project Management – 软工项目管理 · L15

Q7

现假设你和同组的另外 2 位同学，要承担一个新的软件开发项目：为学校图书馆升级现有的图书管理系统，新增加跨馆借阅的功能模块。新增加的需求主要包括：

- 1) 管理员可以配置与其签订合作协议的图书馆信息（如服务访问的 URL 地址、用户名、密码等）；
- 2) 对于签订合作协议的图书馆：
 - a) 用户可以无需登录信息访问其开放的服务接口；
 - b) 用户可以查询其图书信息；
 - c) 用户可以申请跨馆借阅图书；
 - d) 管理员审批跨馆借阅申请，通过后统一办理借阅事宜，并在图书抵达后通知相关申请借阅的用户。

上述项目开发工作须在 1 个月之内完成，请设计你们组基于 Scrum 的迭代开发计划。

A7

- 给出任务，给出开发时间表。具体参考考题B答案。

Process

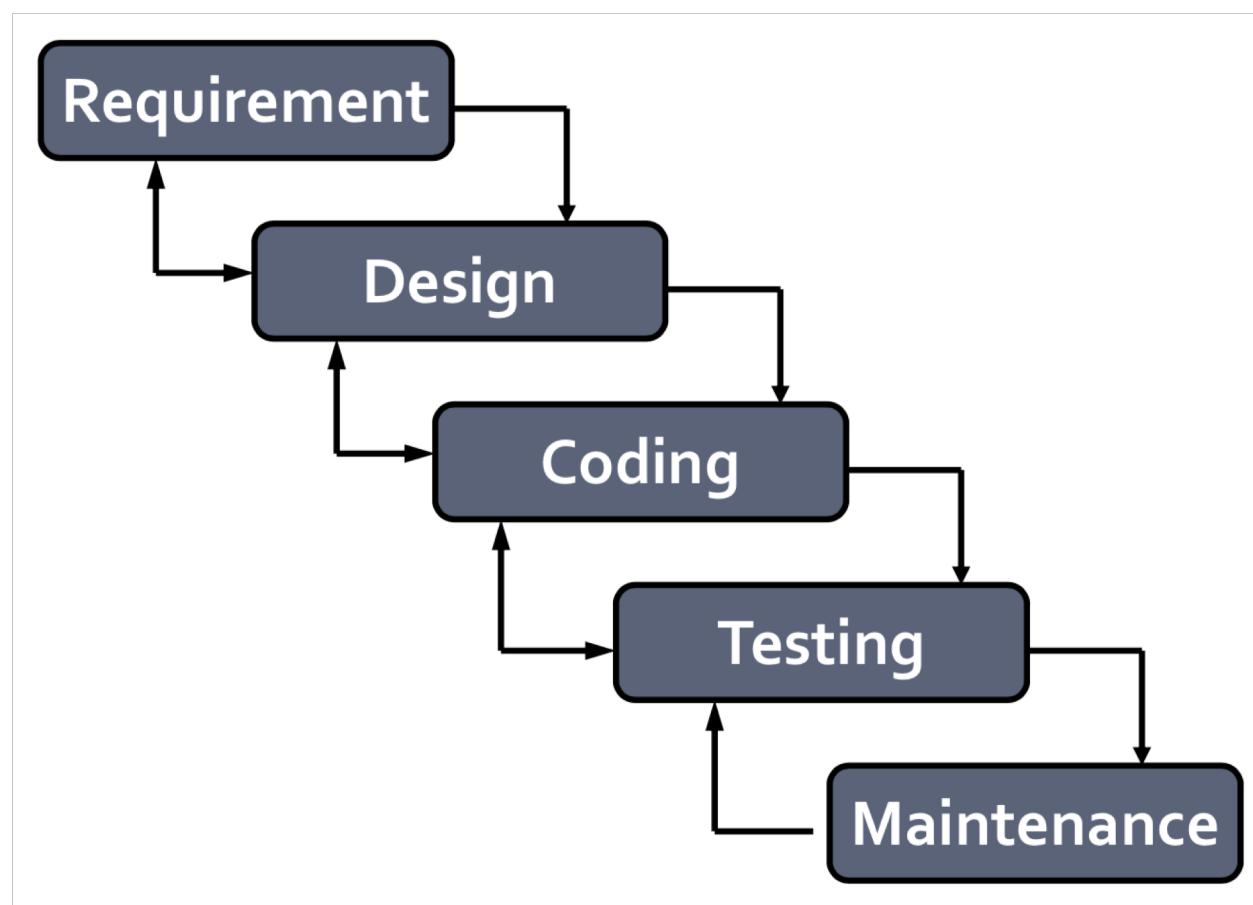
软件生命周期

Process – 软件生命周期 · L14

- 瀑布过程模型 (Waterfall)
- 演化过程模型 (Evolutionary)
 - 增量式开发
 - 原型化开发
 - 螺旋式开发
- 统一软件过程

Process – 软件生命周期 · L14

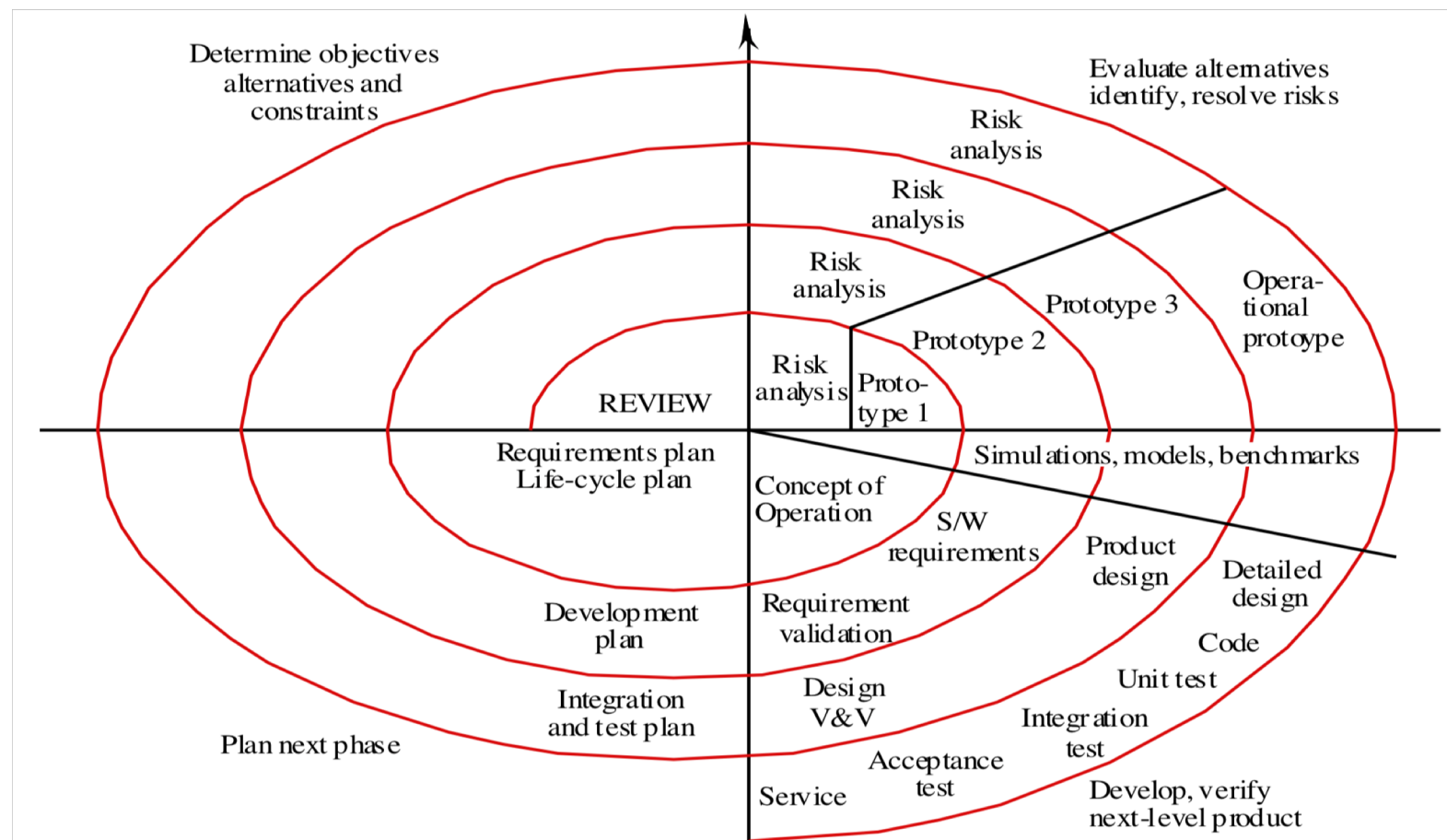
- 瀑布过程模型 (Waterfall)



Process – 软件生命周期 · L14

- 演化过程模型 (Evolutionary)

- 增量式开发
- 原型化开发
- 螺旋式开发



Process – 软件生命周期 · L14

Q8 • 软件开发应该严格遵守需求分析、设计、编码实现、测试的过程；只有在上个阶段完成之后，才能进入下一个阶段的开发工作。

A8 • 1 测试应尽早介入软件开发流程。
• 2 这种模型（Waterfall）的一大问题是流程过于死板、在最开始时很难确定完整的需求、并且只有在测试后才能有一个可以工作的程序。【L14P8】我们目前一般使用增量和迭代开发、一开始就实现基本的四个阶段，或是先做出一个产品的原型（Prototype），再在其上进一步实现需求、改进设计、完善测试。【L14P9-15】

Requirement

需求

Requirement – 需求 · L9-11

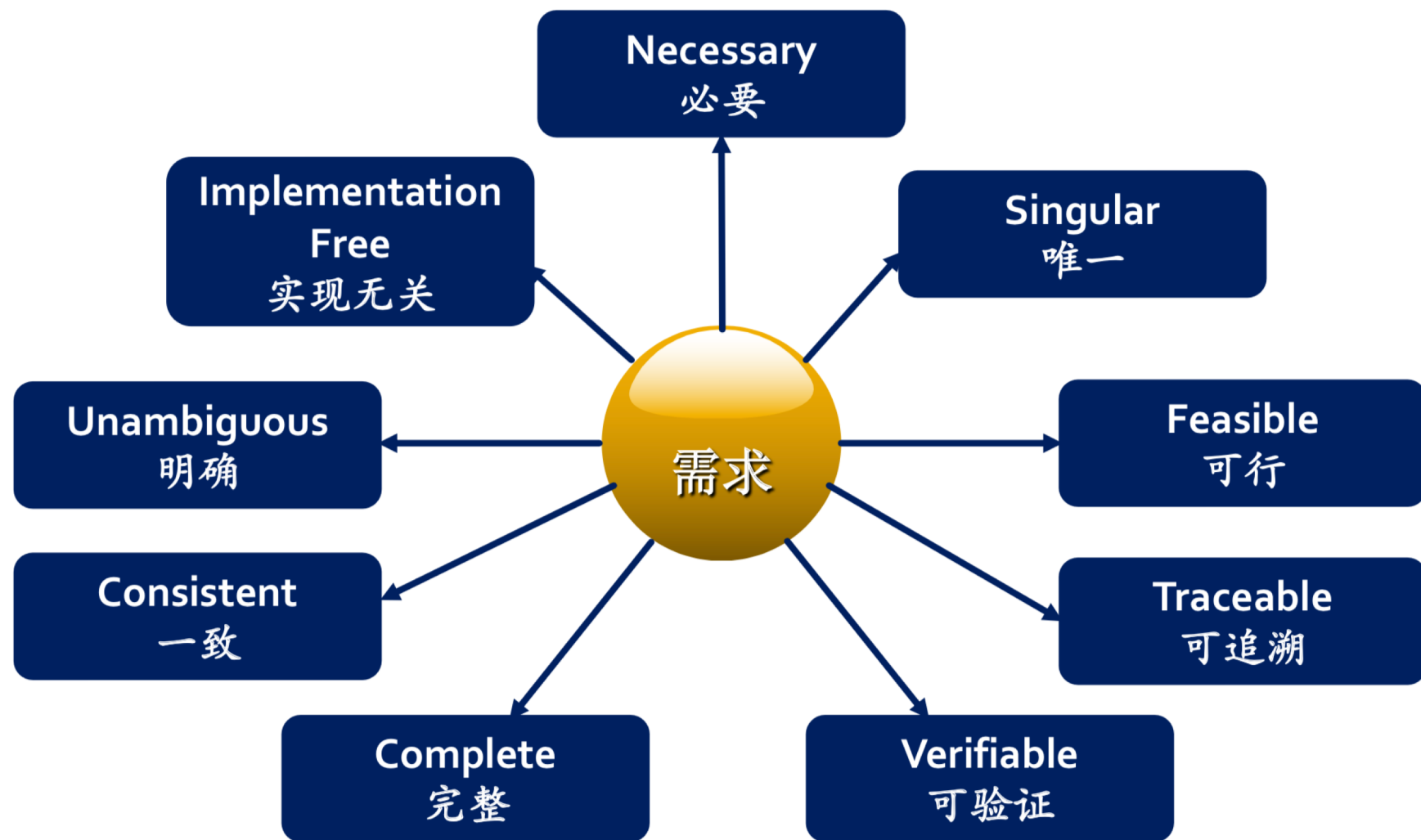
- 需求
 - 画图
 - 评价需求的好坏
 - 设计需求

Requirement – 需求 · L9-11

Q9 • 软件需求要做到完整性和一致性，即C&C。

A9 • 需求需要做到Necessary（必要）、Singular（唯一）、Feasible（可行）、Traceable（可追溯）、Verifiable（可验证）、Complete（完整）、Consistent（一致）、Unambiguous（明确）、Implementation Free（实现无关）
【L11P27】，题目描述不完整。

Requirement – 需求 · L9-11



Requirement – 需求 · L9-11

Q10 • 请对该软件需求进行合理假设，给出 5 个非功能需求的实例

A10 • 写需求的时候，要做到：Necessary（必要）、Singular（唯一）、Feasible（可行）、Traceable（可追溯）、Verifiable（可验证）、Complete（完整）、Consistent（一致）、Unambiguous（明确）、Implementation Free（实现无关）。

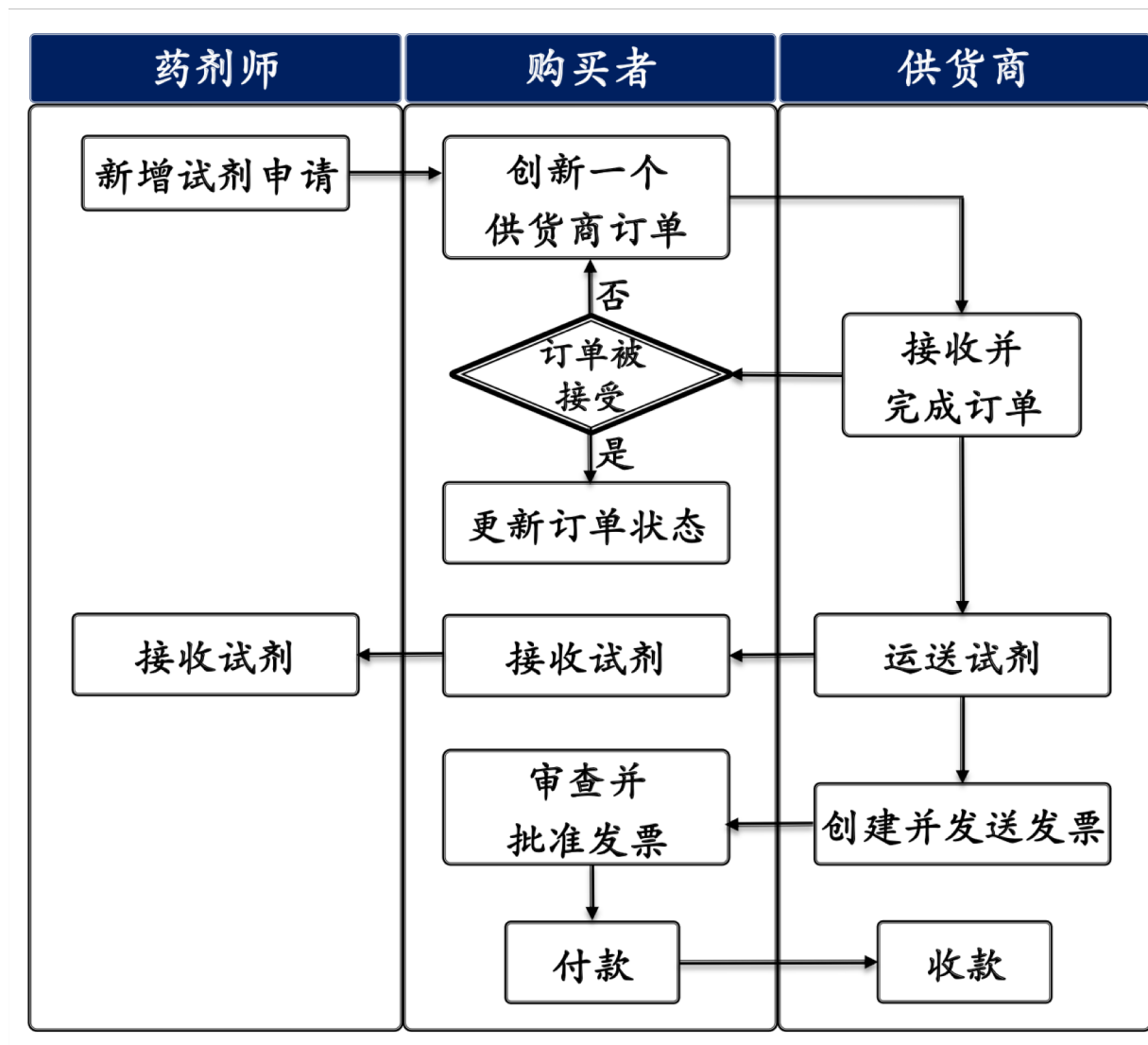
Requirement – 需求 · L9-11

- 画图L10
 - 泳道图 L10P8
 - 数据流图 L10P16
 - 判定树/判定表 L10P30
 - UML类图
 - UML用例图 L10P45
 - UML序列图 L10P66

Requirement – 需求 · L9-11

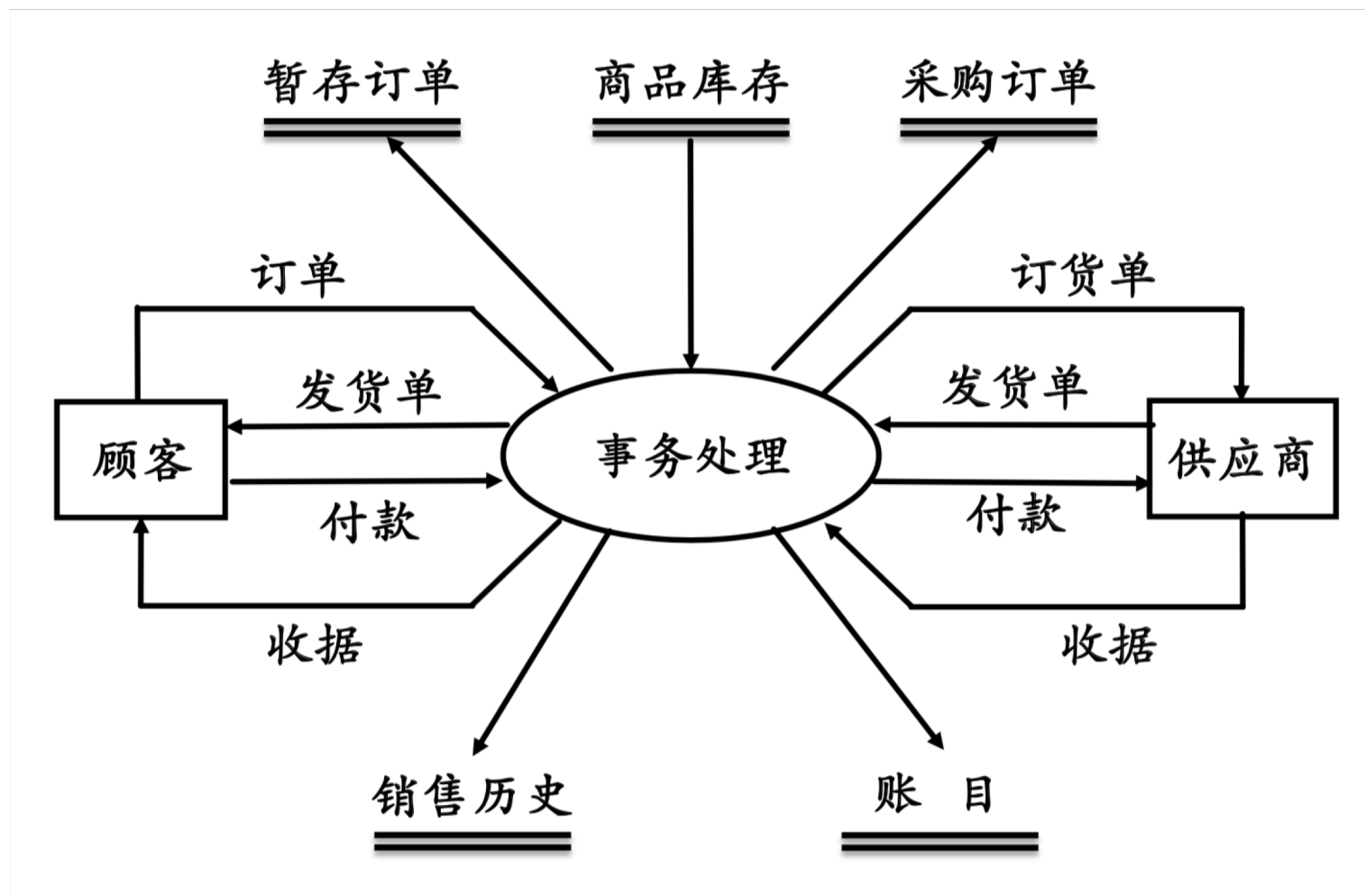
- 泳道图L10P8

- 描述业务过程中涉及的交互过程和操作步骤。



Requirement – 需求 · L9-11

- 数据流图L10P16
 - 描述数据在各个功能模块之间的流动。



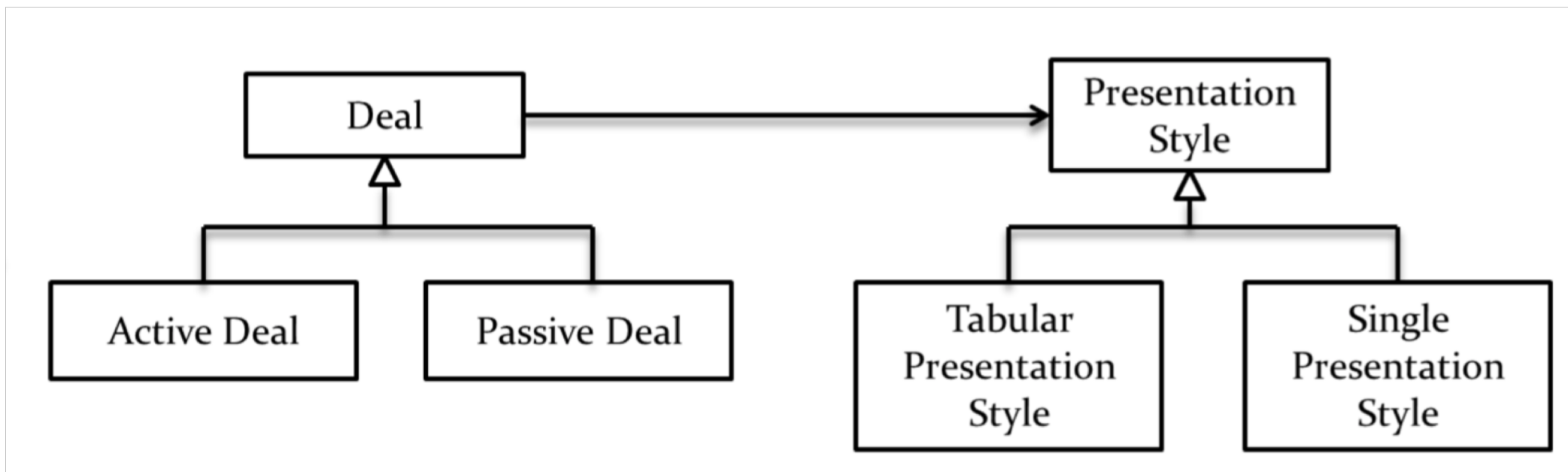
Requirement – 需求 · L9-11

- 判定树/表L10P30
 - 条件组合-系统响应

条件		1	2	3	4	5
C1	用户有权限	F	T	T	T	T
C2	有货	--	F	T	T	T
C3	化学品有毒	--	--	F	T	T
C4	申请人受过培训	--	--	--	F	T
动作						
R1	批准申请			X		X
R2	驳回申请	X	X		X	

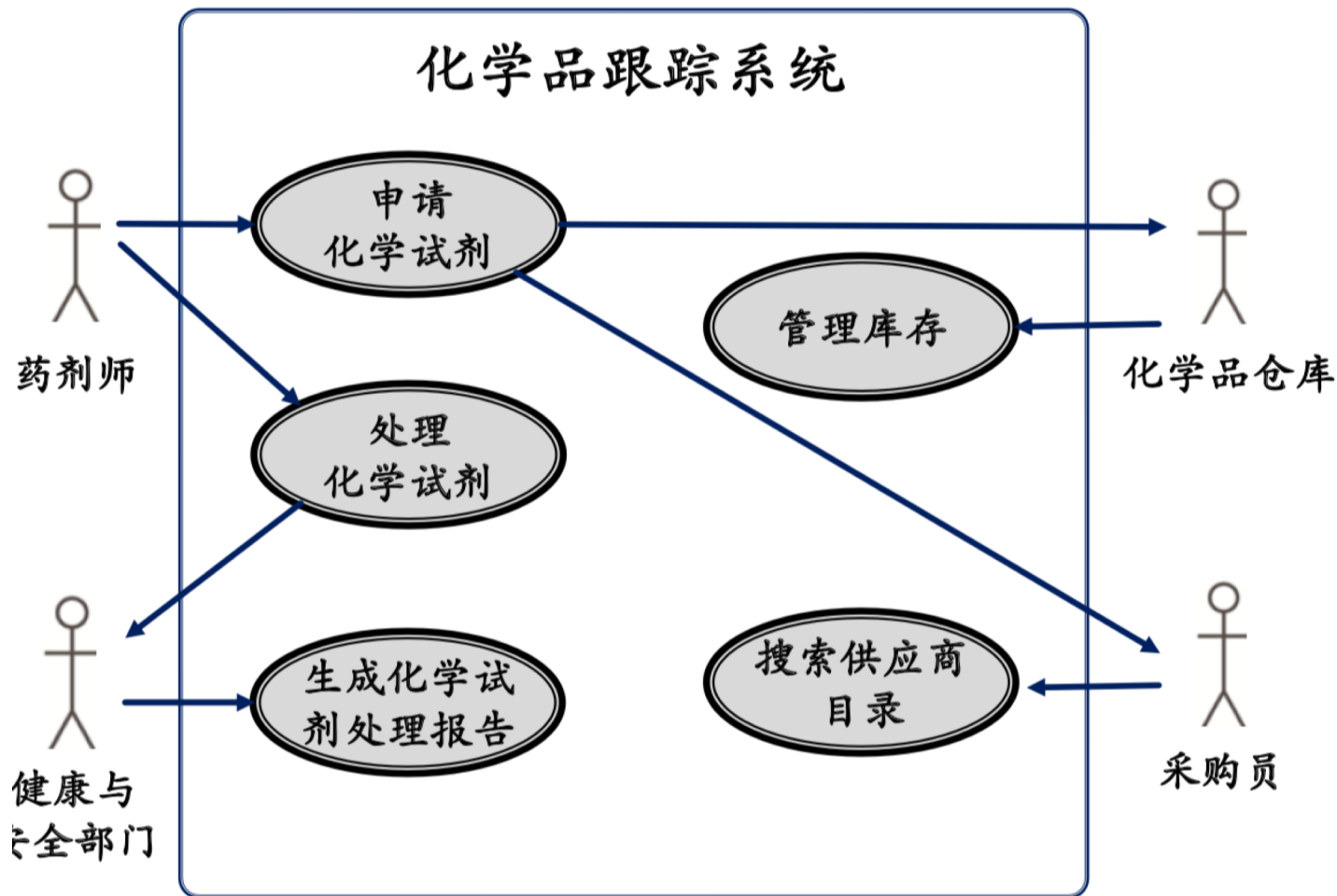
Requirement – 需求 · L9-11

- UML类图
 - 实际上在**设计**阶段
 - OOP
 - 继承派生等



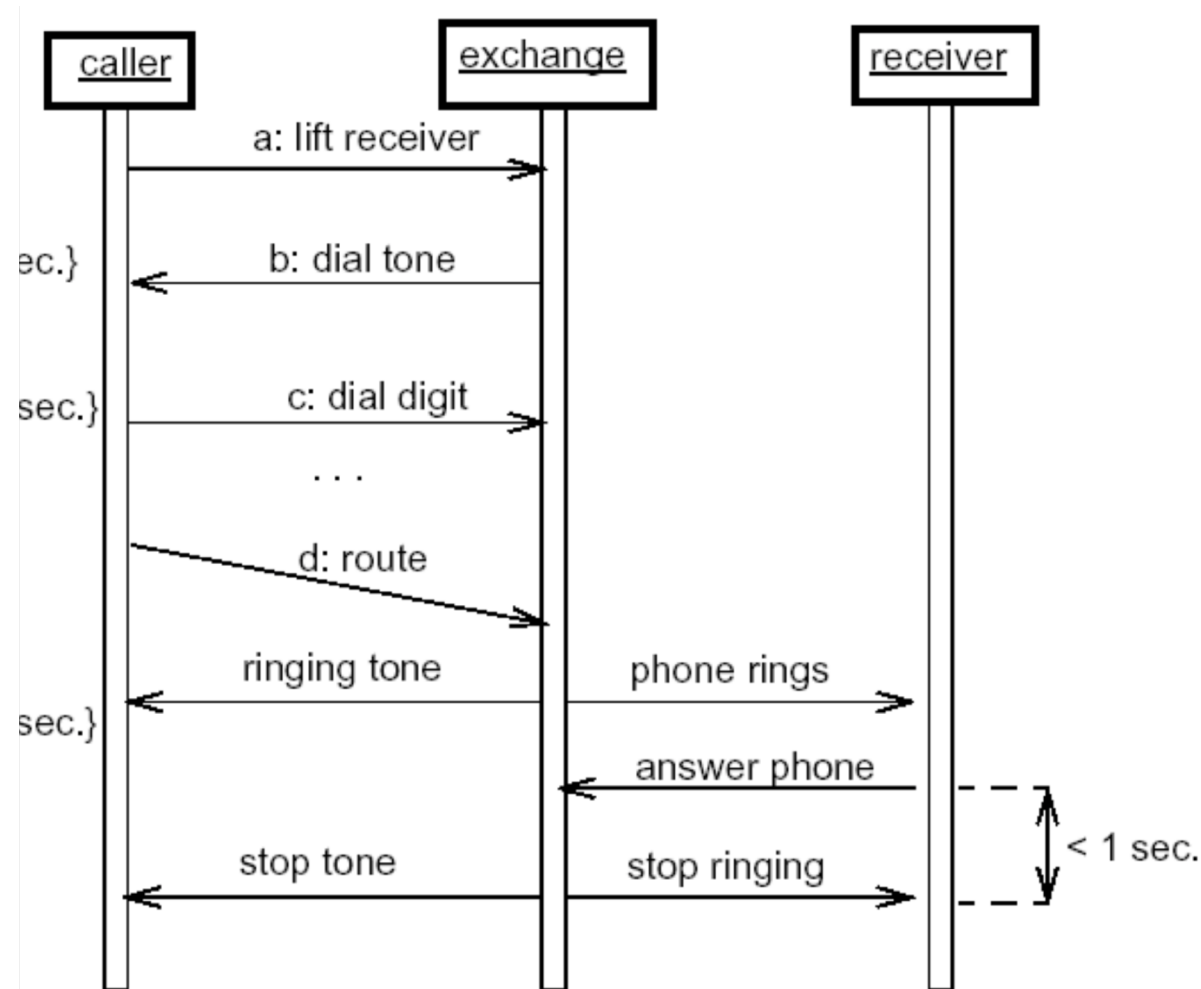
Requirement – 需求 · L9-11

- UML用例图(Use Case) L10P45
 - 系统与某个外部角色之间的一系列交互



Requirement – 需求 · L9-11

- UML序列图
(Sequence) L10P66
 - 展示类是怎么使用的
 - 展示类之间的交互



Example Sequence Diagram

Design

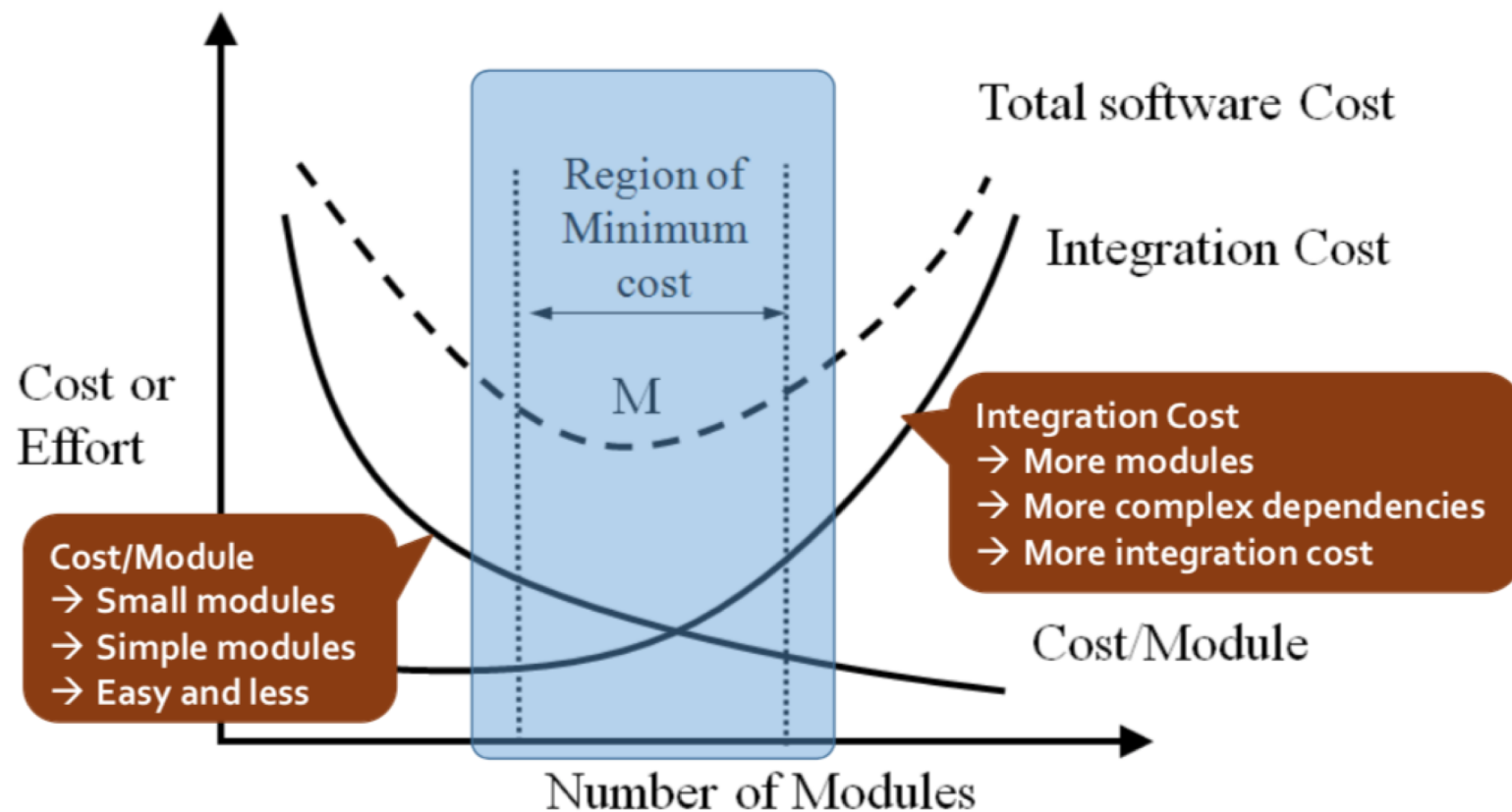
设计

Design – 设计 · L7

- 设计类题目
 - 设计类
 - 设计接口
 - 补全代码
 - 描述缺点/优点

Design – 设计 · L7

Q11



结合图表说明，模块化设计中，软件模块数量及规模对软件开发代价的影响。

Design – 设计 · L7

- A11 • 图中有三条曲线，分别是：开发每一模块的平均代价、集成代价、开发软件总代价。从曲线的走势可以看出，随着模块数量的增加，每一模块会更简单、更小，代价也会不断下降；而模块的增多也意味着更加复杂的依赖关系，从而导致集成代价的不断提升。综合两部分代价，得到的软件的总开发代价是一个U型曲线，故我们需要控制模块的数量，不能太多也不能太少，让曲线的代价控制在U型的低处。

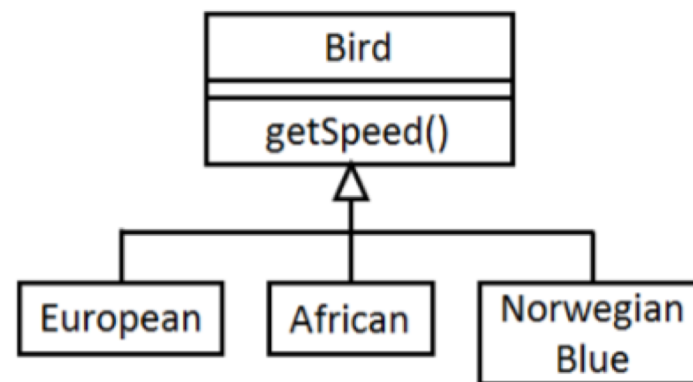
Design – 设计 · L7

Q12

a) 下图中，B 中的类结构是对 A 中代码段的重构，请比较重构前后的设计。

```
double getSpeed () {  
    switch (_type) {  
        case EUROPEAN:  
            return getBaseSpeed ();  
        case AFRICAN:  
            return getBaseSpeed () -  
                getLoadFactor () * _numOfCoconuts;  
        case NORWEGIAN_BLUE:  
            return (_isNailed) ?  
                0: getBaseSpeed (_voltage);  
    }  
    throw new RuntimeException ();  
}
```

A



B

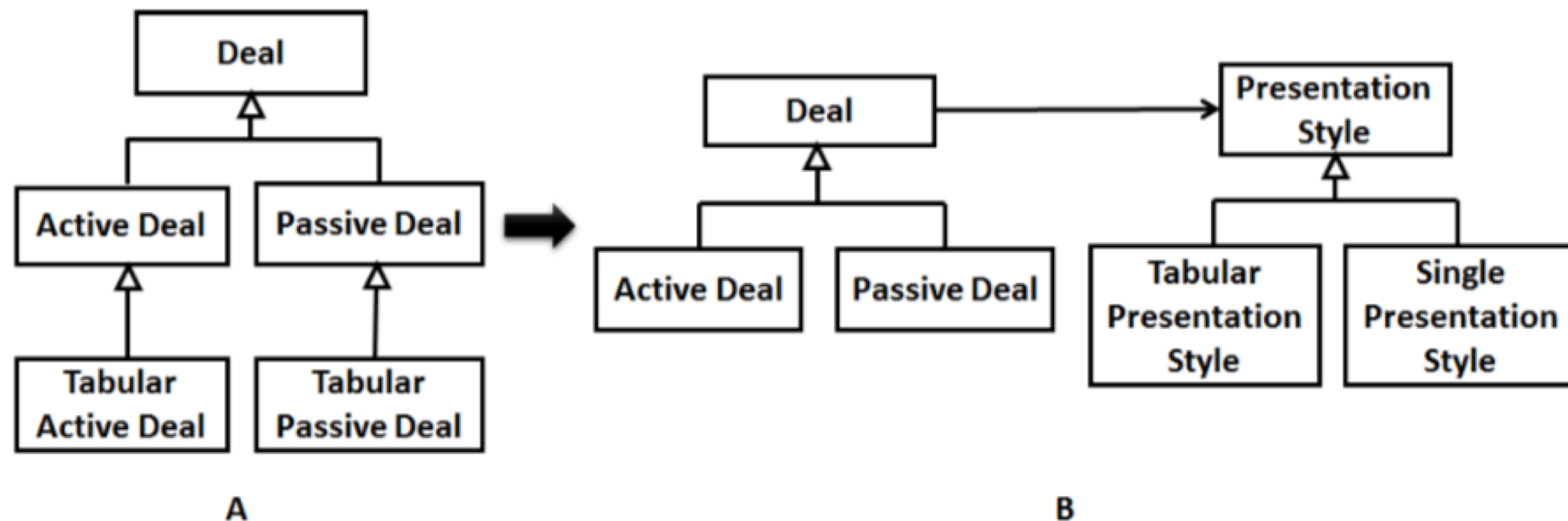
A12

- B的设计让程序的逻辑关系更清晰
- B的设计满足了开闭原则.增加新的类型只需要增加新的代码,而不需要修改已有的实现.

Design – 设计 · L7

Q13

b) 下图中 B 是对 A 的重构，请比较重构前后的设计。



A13

B使用组合代替继承,让逻辑关系更简单.A需要 $m \cdot n$ 个类,B只需要 $m + n$ 个类,实现更简单,避免了类爆炸,提高了代码的可复用性.

Design – 设计 · L7

Q14 • 软件开发应遵循“信息隐藏”的原则，将模块设计成“黑箱”，实现的细节隐藏在模块内部，不让模块的使用者直接访问。这就是信息封装，使用与实现分离的原则。使用者只能通过模块接口访问模块中封装的数据。

A14 • 正确。

Coding

代码实现

Coding – 代码实现 · L5

Q15 • 代码块注释有助于提高程序的可理解性和可维护性。请评审下面两段代码块注释，比较其优劣并分析主要问题。

```
/**
 * Method declaration
 *
 *
 * @param username
 * @param password
 *
 * @return
 *
 * @throws SQLException
 * /
synchronized Channel connect
    (String username, String password)
    throws SQLException {
```

A

```
/**
 * Executes an SQL INSERT, UPDATE, or DELETE statement.
 * In addition, SQL statements that return nothing,
 * such as SQL DDL statements, can be executed.
 *
 * @param sql SQL INSERT, UPDATE, or DELETE statement
 * or a SQL statement that returns nothing @ return
 * either the row count for INSERT, UPDATE, or DELET
 * or 0 for SQL statements that return nothing
 * @exception SQLException if a database access
 * error occurs
 * /
public int executeUpdate (String sql) throws SQLException {
```

B

Coding – 代码实现 · L5

Q16 • 程序设计是一种任人发挥创造才能的技术领域，是个人按自己意图创造的“艺术品”。程序只要能在计算机上得出正确的结果，程序的写法可以不受任何约束。

A16 • 代码要优雅的、可维护的、可复用、可阅读 【L5P3】

Test

测试

Test – 测试 · L4

Q17 • 测试的目的是为了证明程序中没有错误。

A17 • 测试的目的是为了找出软件中的错误。【L4P7】在测试中，我们应该尝试举出大概率能找出错误的测试用例，而不是为了证明程序是正确的。

Test – 测试 · L4

Q18

黑盒测试用来确认系统功能，不考虑程序的内部结构；而白盒测试侧重于程序的结构分析。两种测试方法互为补充。请分别举出至少一个例子，说明：

- 1) 黑盒测试能发现到白盒测试中无法检测到的错误。(3 分)
- 2) 白盒测试能发现到黑盒测试中无法检测到的错误。(3 分)

Test – 测试

A18

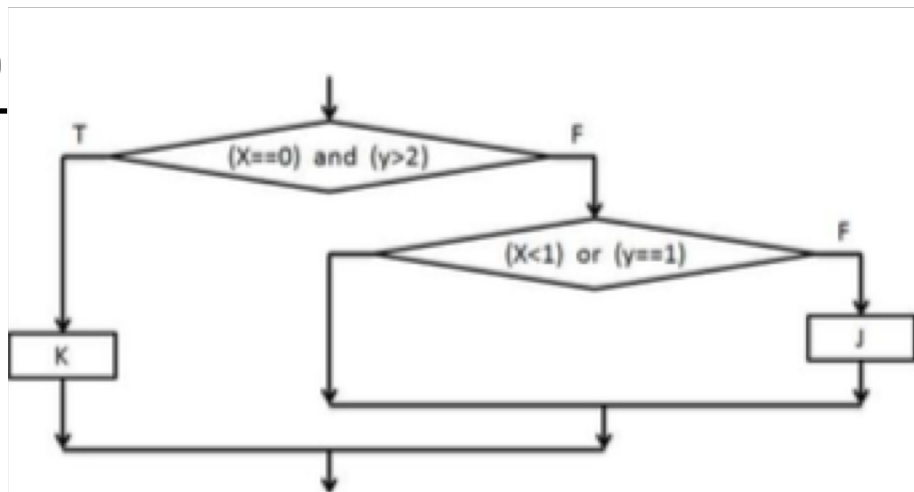
1) 如课件中计算 **Next Date** 的函数，假设函数鲁棒性不强，只对合法输入进行了处理。则在黑盒测试时可用 [2016-2-30]发现“函数对非法输入未做判断”的错误，但白盒测试只能覆盖所有正确输入的情况，无法发现这样的错误。

2) 仍以 **NextDate** 函数为例，假设黑盒测试用等价类测试法，分别测了几组非法输入（如[0-0-32]等）和一种正确输入（[2012-12-12]），未发现问题。但白盒测试中，为覆盖判断二月的分支，输入[2015-2-28]却得到[2015-2-29]，由此才发现代码的错误。

一个更极端的例子，假设在代码中有一段永远不会执行到、但会出错的代码：`while (false) {a = a / 0;}`，则黑盒测试永远无法检测到，但白盒测试在覆盖 `while` 语句的两个分支时则会检测到。

Test – 测试 · L4

Q19



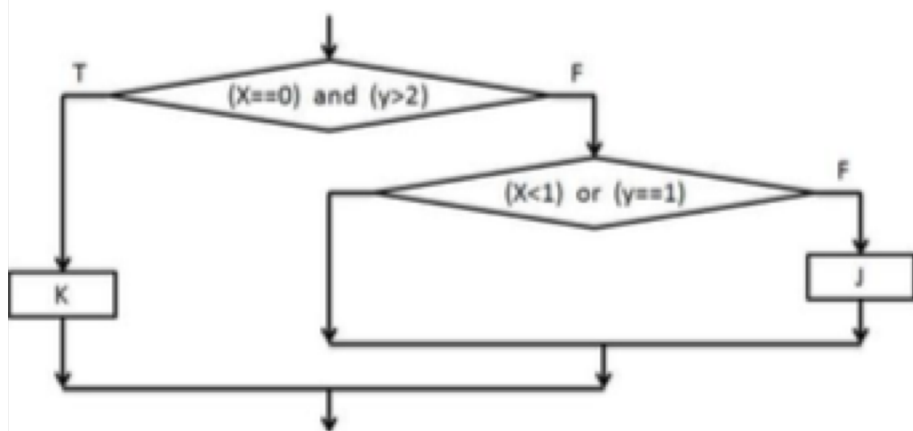
	X	Y
测试用例 I	0	3
测试用例 II	1	2
测试用例 III	-1	2
测试用例 IV	3	1

在结构测试用例设计中，有语句覆盖、条件覆盖、判定覆盖(即分支覆盖)、路径覆盖等。 其中最强的是？

实现判定覆盖至少应采用的测试用例组是 ？， 实现条件覆盖至少应采用的测试用例组是 ？

Test – 测试 · L4

A19



	X	Y
测试用例 I	0	3
测试用例 II	1	2
测试用例 III	-1	2
测试用例 IV	3	1

语句覆盖：执行到J、K

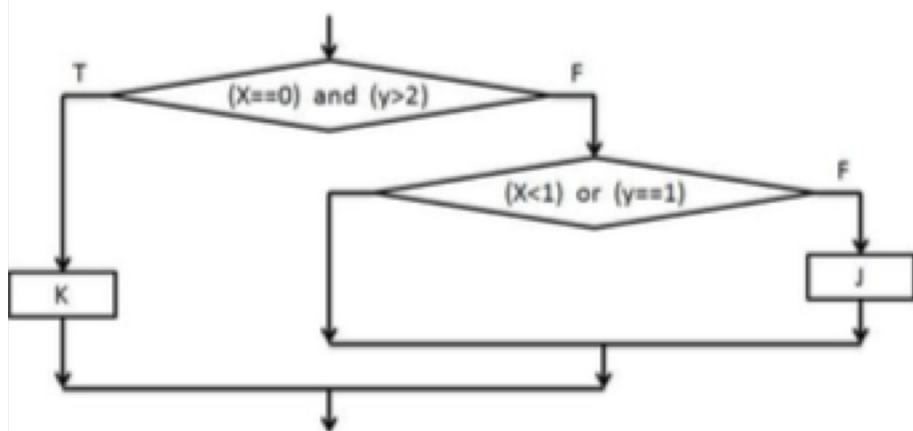
判定（分支）覆盖：每个判断语句的两个分支都取到

条件覆盖：X==0、y>2、x<1、y==1 四个条件的真、假两种情况都取到

路径覆盖：覆盖每条路径至少一次

Test – 测试 · L4

A19



	X	Y
测试用例 I	0	3
测试用例 II	1	2
测试用例 III	-1	2
测试用例 IV	3	1

A: ④路径覆盖 是最强的覆盖准则。

B: 实现判定覆盖至少采用 ⑤I、II 和 III 组

C: 实现条件覆盖至少采用 ④I 和 IV 组

Test – 测试 · L4

Q20

三角形问题是在软件测试文献中使用最广的一个例子。三角形问题接受三个整数 a , b 和 c 作为输入，用做三角形的边。整数 a , b 和 c 必须满足以下条件：

$$c1. \quad 1 \leq a \leq 200$$

$$c2. \quad 1 \leq b \leq 200$$

$$c3. \quad 1 \leq c \leq 200$$

$$c4. \quad a < b + c$$

$$c5. \quad b < a + c$$

$$c6. \quad c < a + b$$

程序的输出是由这三条边确定的三角形类型：等边三角形、等腰三角形、不等边三角形或非三角形，或是错误信息。

- 1) 如果三条边相等，则程序输出“正三角形”。
- 2) 如果任意两边相等，则程序输出“等腰三角形”。
- 3) 如果没有任何两条边相等，则程序输出“不等边三角形”。
- 4) 如果输入值违反了 $c1$, $c2$, $c3$ 中的任何一个，则程序输出错误信息，如“ b 的取值不在允许范围内”。
- 5) 如果输入值违反了 $c4$, $c5$, $c6$ 中的任何一个，则程序输出“非三角形”。

Test – 测试 · L4

Q20

对于上述三角形判定程序，请回答以下问题：

- 1) 任意设计 5 个测试用例，以覆盖上述 5 种不同的输出结果。(5 分)
- 2) 请用等价类方法设计输入数据的等价类，并设计 5 个测试用例。(5 分)
- 3) 请用边界值方法设计 5 个测试用例。(5 分)

Test – 测试 · L4

A20

2)

等价类：对输入空间划分， $c1 \sim c3$ 各对应三个区间 ($x < 1$ 、 $1 \leq x \leq 200$ 、 $x > 200$)， $c4 \sim c6$ 各对应两个区间 (`true`、`false`)，故可以把输入空间划分为 216 个等价类。下面仅取其中 5 个：

Test – 测试 · L4

A20

3) 边界值 (就是对输入区间边界的情况单独测试)

a	b	c	边界情况
1	1	0	超出范围
200	200	200	合法边界
1	1	200	合法边界
200	201	200	超出范围
1	2	3	非三角形

Test – 测试 · L4

Q21

请针对用户登录、存款、取款三项功能，设计测试用例，要求：

- 1) 每个功能设计 5 个测试用例；
- 2) 测试用例设计规范，需包括输入（数据及操作）以及预期输出；
- 3) 测试需要覆盖系统的正常输入和异常输入不同的情况；
- 4) 需采用分区测试和边界值测试方法，且说明测试用例如何覆盖所定义的分区和边界值。

Others

品控、运维

Quality Control – 品控 · L12

Operations – 运维 · L13

总结

总结

- 至少把图表过一遍，看看是什么意思
- 把UML用例图和序列图弄清楚
- 白盒测试和黑盒测试也很重要
- 每一章PPT都找到重点就ok